Guide de l’utilisateur

Amazon EKS

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon EKS: Guide de l’utilisateur

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

..... xxii
..... xxiii
Présentation d'Amazon EKS 1
- Amazon EKS : gestion simplifiée de Kubernetes 1
- Fonctions d'Amazon EKS 2
- Services connexes 4
- Tarification d'Amazon EKS 4
- Cas d'utilisation courants 5
- Architecture 6
  - Plan de contrôle 7
  - Calcul 7
- Concepts de Kubernetes 9
  - Pourquoi Kubernetes ? 10
  - Clusters 15
  - Charges de travail 20
  - Étapes suivantes 26
- Options de déploiement 27
  - Comprendre les options de déploiement d'Amazon EKS 27
  - Amazon EKS dans le cloud 27
  - Amazon EKS dans votre centre de données ou vos environnements périphériques 28
  - Amazon EKS Anywhere pour les environnements isolés 29
  - Outillage Amazon EKS 30

Configuration 31
- Étapes suivantes 31
- Configuration de la AWS CLI 31
  - Pour créer une clé d'accès 32
  - Pour configurer la AWS CLI 32
  - Pour obtenir un jeton de sécurité 33
  - Pour vérifier l'identité de l'utilisateur 33
  - Étapes suivantes 34
- Configurer kubectl et eksctl 34
  - Installation ou mise à jour `kubectl` 34
  - Étape 1 : Vérifiez si elle `kubectl` est installée 35
  - Étape 2 : Installation ou mise à jour `kubectl` 35

Installer `eksctl` ... 49
Étapes suivantes ... 49
Quickstart ... 50
Dans ce tutoriel ... 50
Prérequis ... 51
Configuration du cluster ... 51
Créez IngressClass ... 52
Déployez l'exemple d'application de jeu 2048 ... 52
Persister les données à l'aide du mode automatique d'Amazon EKS ... 56
Supprimer votre cluster et vos nœuds ... 58
Découvrez Amazon EKS ... 59
Présentation ... 59
Atelier Amazon EKS ... 60
Tutoriels pratiques de configuration de clusters Amazon EKS ... 61
Exemples Amazon EKS ... 62
AWS Tutoriels ... 63
Atelier pour développeurs ... 63
Atelier Terraform ... 63
AWS Formation Amazon EKS ... 64
Mise en route ... 65
Créer un cluster (mode automatique EKS) ... 65
Créer un cluster (eksctl) ... 66
Prérequis ... 67
Étape 1 : créer votre cluster Amazon EKS et votre nœud ... 67
Étape 2 : Afficher les ressources Kubernetes ... 68
Étape 3 : supprimer votre cluster et vos nœuds ... 70
Étapes suivantes ... 71
Création d'un cluster (console et CLI) ... 71
Prérequis ... 72
Étape 1 : créer votre cluster Amazon EKS ... 73
Étape 2 : configurer votre ordinateur de façon à ce qu'il communique avec votre cluster ... 76
Étape 3 : créer des nœuds ... 77
Étape 4 : afficher les ressources ... 79
Étape 5 : Suppression des ressources ... 79
Étapes suivantes ... 80
Mode automatique EKS ... 82

Fonctionnalités ... 82
Composants automatisés ... 83
Configuration ... 85
Modèle de responsabilité partagée ... 86
Créer un cluster ... 87
CLI exsctl ... 88
AWS CLI ... 90
Console de gestion ... 98
Activer les clusters existants ... 100
Référence en matière de migration ... 101
Migration de volumes EBS ... 102
Migration des équilibreurs de charge ... 103
Activer sur le cluster ... 104
Migrer depuis Karpenter ... 108
Migrer depuis MNGs ... 111
Migrer depuis Fargate ... 112
Exécuter des charges de travail ... 118
Déployez une charge de travail excessive ... 119
Déployer un équilibreur de charge ... 122
Déployez une charge de travail dynamique ... 127
Configuration ... 133
Créer une classe de nœuds ... 136
Créer un pool de nœuds ... 144
Créer une classe d'entrée ... 151
Créer un service ... 158
Créez StorageClass ... 164
Désactiver le mode automatique EKS ... 175
Mettre à jour la version de Kubernetes ... 176
Passez en revue les pools de nœuds intégrés ... 177
Déploiement du contrôle ... 179
Exécutez des modules complémentaires essentiels ... 181
Utiliser les politiques du réseau ... 183
Sous-réseaux de balises ... 185
Déploiement accéléré de la charge ... 187
Générer un rapport CIS ... 194
Chiffrer les volumes racines ... 197

Mettre à jour les contrôles de l'AMI ... 200
Contrôlez le déploiement de l'ODCR ... 204
Comment ça marche ... 205
Instances gérées ... 206
Identité et accès ... 210
Réseaux ... 216
Dépannage ... 219
Agent de surveillance des nœuds ... 220
Obtenez la sortie de console d'une instance EC2 gérée à l'aide de la AWS EC2 CLI ... 221
Obtenez les journaux des nœuds à l'aide de conteneurs de débogage et de la CLI `kubectl` ... 221
Afficher les ressources associées au mode automatique EKS dans la AWS console ... 222
Afficher les erreurs IAM dans votre AWS compte ... 223
Résoudre les problèmes liés à l'échec de la planification du Pod sur le nœud en mode automatique ... 223
Résoudre les problèmes liés au fait que le nœud ne rejoint pas le cluster ... 224
Partage de volumes entre les pods ... 226
Afficher les événements Karpenter dans les journaux du plan de contrôle ... 227
Résoudre les problèmes liés aux manettes incluses en mode automatique ... 229
Notes de mise à jour ... 229
6 août 2025 ... 229
30 juin 2025 ... 229
20 juin 2025 ... 229
13 juin 2025 ... 230
30 avril 2025 ... 230
18 avril 2025 ... 230
11 avril 2025 ... 230
4 avril 2025 ... 230
31 mars 2025 ... 231
21 mars 2025 ... 231
14 mars 2025 ... 231
Configuration des clusters ... 232
Créer un cluster automatique ... 232
Prérequis ... 51
Créer un cluster - AWS console ... 234
Créer un cluster - AWS CLI ... 239

Étapes suivantes ... 98
Créer un cluster ... 245
Prérequis ... 51
Étape 1 : créer le rôle IAM du cluster ... 246
Étape 2 : créer un cluster ... 248
Étape 3 : Mettre à jour kubeconfig ... 258
Étape 4 : Configuration du cluster ... 259
Étapes suivantes ... 98
Informations sur le cluster ... 260
Types d'informations sur les clusters ... 261
Considérations ... 261
Cas d'utilisation ... 261
Mise en route ... 263
Afficher les informations sur les clusters ... 263
Mettre à jour la version de Kubernetes ... 269
Considérations relatives au mode automatique d'Amazon EKS ... 270
Récapitulatif ... 271
Étape 1 : Préparation de la mise à niveau ... 271
Étape 2 : examiner les considérations relatives à la mise à niveau ... 272
Étape 3 : Mettre à jour le plan de contrôle du cluster ... 273
Étape 4 : Mettre à jour les composants du cluster ... 276
Rétrograder la version Kubernetes pour un cluster Amazon EKS ... 277
Supprimer un cluster ... 278
Considérations ... 103
Supprimer le cluster (eksctl) ... 278
Supprimer le cluster (AWS console) ... 279
Supprimer le cluster (AWS CLI) ... 280
Protégez les clusters EKS contre les suppressions accidentelles ... 282
Accès au point de terminaison de cluster ... 283
`IPv6`format d'extrémité du cluster ... 284
`IPv4`format d'extrémité du cluster ... 284
Point de terminaison privé du cluster ... 285
Modification de l'accès au point de terminaison de cluster ... 286
Accès à un serveur d'API privé uniquement ... 288
Configuration de l'accès aux terminaux ... 289
Activer le support Windows ... 293

Considérations ... 103
Prérequis ... 51
Activer le support Windows ... 295
Déployer des modules Windows ... 297
Support d'une densité de pods plus élevée sur les nœuds Windows ... 298
Désactiver le support Windows ... 298
Clusters privés ... 299
Autoscaling ... 303
Mode automatique EKS ... 303
Solutions supplémentaires ... 303
En savoir plus sur Zonal Shift ... 304
Comprendre le flux de trafic réseau est-ouest entre les modules ... 305
Exigences d'EKS Zonal Shift ... 310
Questions fréquentes (FAQ) ... 319
Ressources supplémentaires ... 322
Activer le changement de zone ... 322
Considérations ... 323
Qu'est-ce qu'Amazon Application Recovery Controller ? ... 323
Qu'est-ce que le décalage zonal ? ... 323
Qu'est-ce que l'autoshift zonal ? ... 324
Que fait EKS lors d'un changement automatique ? ... 324
Enregistrer le cluster EKS auprès d'Amazon Application Recovery Controller (ARC) (AWS console) ... 324
Étapes suivantes ... 98
Gérez l'accès ... 326
Tâches communes ... 326
Contexte ... 185
Considérations relatives au mode automatique EKS ... 278
Accès à l'API Kubernetes ... 328
Associer les identités IAM aux autorisations Kubernetes ... 329
Définir le mode d'authentification du cluster ... 330
Accorder des autorisations ... 331
aws-auth ConfigMap ... 375
Lien vers le fournisseur OIDC ... 385
Dissocier le fournisseur OIDC ... 390
Accès aux ressources du cluster ... 390

Autorisations requises ... 392
Cluster d'accès avec kubectl ... 397
Créer le fichier `kubeconfig` automatiquement ... 399
Accès à la charge de travail à AWS ... 400
Jetons de compte de service ... 400
Modules complémentaires de cluster ... 402
Octroi AWS d'autorisations Identity and Access Management aux charges de travail sur les clusters Amazon Elastic Kubernetes Service ... 402
Identifiants auprès de l'IRSA ... 408
Identité du pod ... 433
Gérer le calcul ... 468
Comparez les options de calcul ... 468
Groupes de nœuds gérés ... 476
Concepts des groupes de nœuds gérés ... 477
Types de capacité des groupes de nœuds gérés ... 479
Création ... 482
Mettre à jour ... 494
Mettre à jour les détails du comportement ... 500
Modèles de lancement ... 504
Suppression ... 522
Nœuds autogérés ... 524
Amazon Linux ... 525
Bottlerocket ... 535
Windows ... 539
Ubuntu Linux ... 549
Méthodes de mise à jour ... 552
AWS Fargate ... 566
AWS Considérations relatives à Fargate ... 566
Tableau comparatif de Fargate ... 569
Mise en route ... 572
Définir des profils ... 578
Supprimer des profils ... 583
Détails de configuration du pod ... 584
Événements relatifs aux correctifs du système d'exploitation ... 587
Collectez des métriques ... 590
Journalisation ... 592

Types d' EC2 instances Amazon 604
Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon 607
Considérations relatives au mode automatique EKS 278
Pré-construit optimisé AMIs 608
AL2 Obsolète de l'AMI 609
Amazon Linux 615
Bottlerocket 625
Ubuntu Linux 631
Windows 631
Santé des nœuds 733
Agent de surveillance des nœuds 733
Réparation automatique de nœuds 734
Problèmes de santé des nœuds 735
Afficher l'état du nœud 747
Obtenir les journaux des nœuds 749
Nœuds hybrides 752
Fonctionnalités 753
Limites 754
Considérations 754
Ressources supplémentaires 755
Prérequis 755
Exécutez des nœuds hybrides 826
Configuration 848
Comment ça marche 889
Nœuds hybrides nodeadm 930
Résolution des problèmes 947
Stockage des données des applications 969
Amazon EBS 969
Considérations 970
Prérequis 970
Étape 1 : Créer un rôle IAM 971
Étape 2 : obtenir le pilote Amazon EBS CSI 979
Étape 3 : Déployer un exemple d'application 979
Amazon EFS 979
Considérations 980

Prérequis 980
Étape 1 : Créer un rôle IAM 981
Étape 2 : obtenir le pilote Amazon EFS CSI 987
Étape 3 : Créer un système de fichiers Amazon EFS 988
Étape 4 : Déployer un exemple d'application 988
Amazon FSx pour Lustre 988
Déployez le pilote 988
Optimiser (EFA) 997
Optimize (non-EFA) 1013
Amazon FSx pour NetApp ONTAP 1021
Amazon FSx pour OpenZFS 1022
Cache de fichiers Amazon 1022
Mountpoint pour Amazon S3 1023
Considérations 1023
Déployez le pilote 1024
Supprimer le pilote 1034
Contrôleur d'instantané CSI 1036
Configuration de la mise en réseau 1037
Ajouter un sous-réseau VPC existant à un cluster Amazon EKS depuis la console de gestion 1037
Exigences requises pour le VPC et les sous-réseaux 1038
Exigences et considérations requises pour le VPC 1038
Exigences et considérations requises pour les sous-réseaux 1040
Exigences et considérations requises pour les sous-réseaux partagés 1047
Création d'un VPC 1048
Prérequis 51
Sous-réseaux publics et privés 1049
Sous-réseaux publics uniquement 1051
Sous-réseaux privés uniquement 1053
Exigences de groupe de sécurité 1055
Groupe de sécurité du cluster par défaut 1055
Restriction du trafic de cluster 1057
Groupes de sécurité partagés 1058
Gérer les modules complémentaires réseau 1059
Modules complémentaires intégrés 1059
Compléments AWS réseau optionnels 1060
CNI Amazon VPC 1061

Plugins CNI alternatifs 1183
Multus 1185
AWS Contrôleur Load Balancer 1186
CoreDNS 1205
kube-proxy 1225
Charges de travail 1231
Exemple de déploiement (Linux) 1231
Prérequis 51
Création d'un espace de noms 1232
Création d'un déploiement Kubernetes 1232
Créer un service 1234
Réviser les ressources créées 1235
Exécuter un shell sur un Pod 1238
Étapes suivantes 1239
Exemple de déploiement (Windows) 1239
Prérequis 51
Création d'un espace de noms 1232
Création d'un déploiement Kubernetes 1232
Créer un service 1234
Réviser les ressources créées 1243
Exécuter un shell sur un Pod 1238
Étapes suivantes 1247
Vertical Pod Autoscaler (VPA) 1247
Déploiement du VPA (Vertical Pod Autoscaler) 1247
Test de l'installation de votre VPA (Vertical Pod Autoscaler) 1249
Horizontal Pod Autoscaler 1253
Exécution d'une application de test du Horizontal Pod Autoscaler 1254
Répartition de charge réseau 1256
Prérequis 51
Considérations 103
Créer un équilibreur de charge de réseau 1261
(Facultatif) Déployer un exemple d'application 1263
Répartition de la charge des applications 1267
Prérequis 51
Réutilisation ALBs avec des groupes d'entrée 1270
(Facultatif) Déployer un exemple d'application 1272

Restreindre le service externe IPs 1275
Copier une image dans un référentiel 1277
Afficher les registres d'images Amazon 1281
Modules complémentaires Amazon EKS 1284
Considérations 1285
Considérations relatives au mode automatique d'Amazon EKS 1286
Support 1287
AWS modules complémentaires 1289
Modules complémentaires communautaires 1308
Modules complémentaires Marketplace 1314
Création d'un module complémentaire 1331
Mettre à jour un module complémentaire 1345
Vérifiez la compatibilité 1353
Supprimer un module complémentaire 1355
Rôles IAM 1359
Champs que vous pouvez personnaliser 1366
Vérification d'images de conteneur 1370
Gestion du cluster 1371
Suivi des coûts 1371
Afficher les coûts par module 1372
Installez Kubecost 1373
Accédez au tableau de bord Kubecost 1374
En savoir plus sur Kubecost 1375
Serveur de métriques 1386
Considérations 103
Déployez en tant que module complémentaire communautaire avec les modules complémentaires Amazon EKS 1386
Déployer avec un manifeste 1387
Déployez des applications avec Helm 1388
Balisage de vos ressources 1390
Principes de base des étiquettes 1390
Balisage de vos ressources 1391
Restrictions liées aux étiquettes 1392
Identification de vos ressources pour facturation 1393
Gestion des étiquettes à l'aide de la console 1394
Gestion des identifications à l'aide de la CLI, de l'API ou de `eksctl` 1395

Quotas de service ... 1396
Consultez les quotas de service EKS dans le AWS Management Console ... 1397
Afficher les quotas de service EKS avec la AWS CLI ... 1397
Service quotas Amazon EKS ... 1398
AWS Quotas de service Fargate ... 1399
Sécurité ... 1400
Bonnes pratiques ... 1401
Analyser les vulnérabilités ... 1401
Le test de référence du Center for Internet Security (CIS) pour Amazon EKS ... 1402
Versions de la plateforme Amazon EKS ... 1402
Liste des vulnérabilités du système d'exploitation ... 1403
Détection de nœuds avec Amazon Inspector ... 1403
Détection de clusters et de nœuds avec Amazon GuardDuty ... 1403
Valider la conformité ... 1403
Considérations relatives à EKS ... 1405
Sécurité de l'infrastructure ... 1405
Résilience ... 1410
Considérations relatives à Kubernetes ... 1411
Signature des certificats ... 1411
Rôles et utilisateurs par défaut ... 1413
Activer le chiffrement secret ... 1418
AWS Secrets Manager ... 1423
Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes ... 1424
Considérations relatives à EKS Auto ... 1432
Sécurité et authentification des API ... 1433
Sécurité du réseau ... 1433
EC2 sécurité des instances gérées ... 1434
Gestion automatisée des ressources ... 1435
Bonnes pratiques de sécurité ... 1436
Référence IAM ... 1436
Public ciblé ... 1437
Authentification par des identités ... 1437
Gestion des accès à l'aide de politiques ... 1441
Amazon EKS et IAM ... 1443
Politiques basées sur l'identité ... 1448
Rôles liés à un service ... 1456

Rôle IAM d'exécution de pod ... 1473
Rôle IAM du connecteur ... 1478
AWS politiques gérées ... 1482
Résolution des problèmes ... 1509
Rôle IAM du cluster ... 1512
Rôle IAM de nœud ... 1516
Rôle IAM du cluster en mode automatique ... 1522
Rôle IAM du nœud en mode automatique ... 1525
Surveiller les clusters ... 1529
Surveillance et journalisation sur Amazon EKS ... 1530
Outils de surveillance et de journalisation Amazon EKS ... 1531
Tableau de bord d'observabilité ... 1535
Récapitulatif ... 1535
Santé du cluster ... 1535
Surveillance du plan de contrôle ... 1536
Informations sur le cluster ... 1538
Problèmes de santé des nœuds ... 1539
Métriques Prometheus ... 1539
Étape 1 : activer les métriques Prometheus ... 1540
Étape 2 : Utiliser les métriques Prometheus ... 1542
Étape 3 : Gérez les grattoirs Prometheus ... 1542
Déployer avec Helm ... 1543
Plan de contrôle ... 1546
Amazon CloudWatch ... 1554
Statistiques de base sur Amazon CloudWatch ... 1554
Opérateur Amazon CloudWatch Observability ... 1563
Journaux du plan de contrôle ... 1564
Activer ou désactiver les journaux du plan de contrôle ... 1565
Afficher les journaux du plan de contrôle du cluster ... 1568
AWS CloudTrail ... 1569
Références ... 1570
Entrées du fichier journal ... 1571
Métriques du groupe Auto Scaling ... 1574
Opérateur ADOT ... 1574
Tableau de bord Amazon EKS ... 1575
Qu'est-ce que le tableau de bord Amazon EKS ? ... 1575

Comment le tableau de bord utilise-t-il AWS les Organizations ? 1575
Activez le tableau de bord EKS à l'aide de la AWS console 1577
Afficher le tableau de bord EKS 1578
Configuration du tableau de bord 1578
Désactiver le tableau de bord EKS à l'aide de la AWS console 1579
Résoudre les problèmes liés au tableau de bord EKS 1580
Configuration des organisations 1580
Utilisation avec d'autres services 1587
AWS CloudFormation 1587
Amazon EKS et AWS CloudFormation modèles 1587
En savoir plus sur AWS CloudFormation 1588
Amazon Detective 1588
Utilisation d'Amazon Detective avec Amazon EKS 1589
Amazon GuardDuty 1589
AWS Centre de résilience 1591
Amazon Security Lake 1591
Avantages de l'utilisation de Security Lake avec Amazon Amazon EKS 1592
Activation de Security Lake pour Amazon EKS 1592
Analyse des journaux EKS dans Security Lake 1593
Amazon VPC Lattice 1593
AWS Zones Locales 1593
Amazon EKS Connector 1595
Considérations relatives à Amazon EKS Connector 1595
Rôles IAM requis pour Amazon EKS Connector 1596
Connecter un cluster 1596
Considérations 1597
Prérequis 1597
Étape 1 : Enregistrement du cluster 1597
Étape 2 : Installation de l'agent `eks-connector` 1600
Étapes suivantes 1602
Accorder l'accès aux clusters 1602
Prérequis 1603
Annuler l'enregistrement d'un cluster 1604
Désenregistrer le cluster Kubernetes 1605
Nettoyez les ressources de votre cluster Kubernetes 1606
Résoudre les problèmes liés au connecteur EKS 1606

Dépannage de base ... 1607
Problème Helm : 403 Forbidden ... 1608
Erreur de console : le cluster est bloqué dans l'état En attente ... 1609
Erreur de console : l'utilisateur system:serviceaccount:eks-connector:eks-connector ne peut pas se faire passer pour les utilisateurs de ressources du groupe d'API à la portée du cluster ... 1609
Erreur de console : [...] est interdit : l'utilisateur [...] ne peut pas répertorier la ressource [...] dans le groupe d'API à l'échelle du cluster ... 1610
Erreur de console : Amazon EKS ne peut pas communiquer avec le serveur API de votre cluster Kubernetes. Le cluster doit être dans un état ACTIF pour que la connexion soit réussie. Veuillez réessayer dans quelques minutes. ... 1610
Les pods du connecteur Amazon EKS fonctionnent en boucle ... 1611
Impossible de lancer le connecteur EKS : InvalidActivation ... 1611
Le nœud du cluster manque de connectivité sortante ... 1612
Les pods du connecteur Amazon EKS sont en bon `ImagePullBackOff` état ... 1613
Questions fréquentes (FAQ) ... 1613
Considérations sur la sécurité ... 1614
AWS responsabilités ... 1615
Responsabilités client ... 1615
Amazon EKS sur AWS Outposts ... 1617
Quand utiliser chaque option de déploiement ... 1617
Comparaison des options de déploiement ... 1618
Exécuter des clusters locaux ... 1621
AWS Régions prises en charge ... 1621
Déployer un cluster local ... 1621
Versions de la plateforme EKS ... 1633
Créer un VPC et des sous-réseaux ... 1647
Préparez-vous aux déconnexions ... 1651
Considérations relatives à la capacité ... 1656
Résolution des problèmes liés à un cluster ... 1659
Nœuds ... 1671
`eksctl` ... 1672
AWS Management Console ... 1674
AI/ML sur EKS ... 1681
Pourquoi choisir EKS pour l'IA et le ML ? ... 1681
Principaux cas d'utilisation ... 1682

Études de cas 1683
Commencez à utiliser le Machine Learning sur EKS 1684
Inférence en temps réel 1684
Créer un cluster 1684
Configuration de cluster 1722
Configurer le GPU Linux AMIs 1722
Configurer le GPU Windows AMIs 1725
Configurez des clusters de formation avec EFA 1731
Configurer des clusters d'inférence avec Inferentia 1743
Gestion de capacité 1749
Réserve GPUs pour les groupes de nœuds gérés 1750
Réserve GPUs pour les nœuds autogérés 1753
Recettes 1756
Empêcher la planification des pods sur des nœuds spécifiques 1757
Ressources 1759
Ateliers 1759
Bonnes pratiques 1760
Architectures de référence 1761
Didacticiels 1762
Gestion des versions 1764
Versions Kubernetes 1764
Versions disponibles bénéficiant du support standard 1765
Versions disponibles bénéficiant du support étendu 1765
Calendrier de publication Amazon EKS Kubernetes 1765
Obtenir des informations de version avec la AWS CLI 1766
Version d'Amazon EKS FAQs 1768
Support étendu d'Amazon EKS FAQs 1770
Versions bénéficiant du support standard 1773
Versions bénéficiant d'un support étendu 1779
Afficher la période de support 1782
Afficher la politique de mise à niveau 1783
Activer le support étendu 1785
Désactiver le support étendu 1786
Versions de plateforme 1787
Version de Kubernetes 1.33 1789
Version de Kubernetes 1.32 1790

Version de Kubernetes 1.31 ... 1793
Version de Kubernetes 1.30 ... 1797
Version de Kubernetes 1.29 ... 1802
Version de Kubernetes 1.28 ... 1808
Version de Kubernetes 1.27 ... 1816
Version de Kubernetes 1.26 ... 1823
Obtenir la version actuelle de la plateforme ... 1831
Modifier la version de la plateforme ... 1832
Résolution des problèmes ... 1833
Capacité insuffisante ... 1833
Les nœuds ne parviennent pas à joindre le cluster ... 1833
Accès non autorisé ou refusé (`kubectl`) ... 1835
`hostname doesn't match` ... 1836
`getsockopt: no route to host` ... 1837
`Instances failed to join the Kubernetes cluster` ... 1837
Codes d'erreurs liées aux groupes de nœuds gérés ... 1837
`Not authorized for images` ... 1842
Le nœud est en `NotReady` état ... 1842
Collecteur de journaux EKS ... 1843
Le réseau d'exécution du conteneur n'est pas prêt ... 1844
Délai d'expiration de la liaison TLS ... 1845
InvalidClientTokenId ... 1846
Les groupes de nœuds doivent correspondre à la version de Kubernetes avant de mettre à niveau le plan de contrôle ... 1847
Lors du lancement de nombreux nœuds, des erreurs `Too Many Requests` se produisent ... 1847
Réponse d'erreur non autorisée HTTP 401 sur les requêtes du serveur API Kubernetes ... 1847
La version de la plateforme Amazon EKS est inférieure de plus de deux versions à la version actuelle de la plateforme ... 1848
Codes d'erreur FAQs et d'intégrité du cluster avec chemins de résolution ... 1851
Projets liés à Amazon EKS ... 1858
Support pour les logiciels déployés sur EKS ... 1858
Outils de gestion ... 1859
eksctl ... 1859
AWS contrôleurs pour Kubernetes ... 1859
Flux CD ... 1860
CDK pour Kubernetes ... 1860

Réseaux ... 1860
Plugin CNI Amazon VPC pour Kubernetes ... 1860
AWS Contrôleur Load Balancer pour Kubernetes ... 1860
ExternalDNS ... 1861
Machine learning ... 1861
Kubeflow ... 1861
Auto Scaling ... 1861
Cluster Autoscaler ... 1861
Karpenter ... 1862
Escalator ... 1862
Surveillance ... 1862
Prometheus ... 1862
Intégration continue/déploiement continu ... 1862
Jenkins X ... 1863
Nouvelles fonctionnalités et feuille de route ... 1864
Historique du document ... 1865
Contribuer ... 1927
Modifier une seule page ... 1927
Prérequis ... 51
Procédure ... 106
Vue d'ensemble de la pull request ... 1929
Modifiez des fichiers avec GitHub ... 1930
Prérequis ... 51
Procédure ... 106
Afficher les commentaires sur le style ... 1931
Installez Vale ... 1932
Installez l'extension VS Code Vale ... 1932
Valeur de synchronisation ... 1933
Afficher les commentaires de style dans VS Code ... 1933
Créer une page ... 1933
Créer une page ... 1934
Ajouter une page à la navigation ... 1934
Insérer un lien ... 1935
Lien vers une page ou une section du guide de l'utilisateur EKS ... 1935
Lien vers un autre guide dans les AWS documents ... 1935
Lien vers une page Web externe ... 1936

Créez avec Amazon Q 1936
Installez Amazon Q avec VS Code 1937
Connectez-vous à Amazon Q 1937
Utiliser Amazon Q pour créer du contenu 1938
Conseils 1938
Afficher l'aperçu des relations publiques 1938
Afficher l'aperçu 1939
Limitations propres à la version préliminaire 1940
AsciiDoc syntaxe 1940
Nouvelle page 1941
En-têtes 1941
Formatage du texte 1941
Listes 1941
Liens 1942
Exemples de code 1942
Images 1942
Tables 1942
Alertes 1943
Inclure d'autres fichiers 1943
Attributs (similaires aux entités) 1943
Procédures 1944

Aidez à améliorer cette page

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.

Pour contribuer à ce guide de l'utilisateur, cliquez sur le GitHub lien Modifier cette page sur qui se trouve dans le volet droit de chaque page.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.

Présentation d'Amazon EKS

Amazon EKS : gestion simplifiée de Kubernetes

Amazon Elastic Kubernetes Service (EKS) fournit un service Kubernetes entièrement géré qui élimine la complexité liée à l'exploitation de clusters Kubernetes. Avec EKS, vous pouvez :

- Déployez les applications plus rapidement avec moins de frais opérationnels
- Évoluez facilement pour répondre à l'évolution des demandes de charge de travail
- Améliorez la sécurité grâce à AWS l'intégration et aux mises à jour automatisées
- Choisissez entre le mode EKS standard ou le mode automatique EKS entièrement automatisé

Amazon Elastic Kubernetes Service (Amazon EKS) est la première plateforme pour exécuter des clusters Kubernetes, à la fois dans le cloud Amazon Web Services () et dans vos propres centres de données (EKS Anywhere et Amazon EKS Hybrid Nodes).AWS

Amazon EKS simplifie la création, la sécurisation et la maintenance des clusters Kubernetes. Il peut être plus rentable de fournir suffisamment de ressources pour répondre à la demande de pointe que de gérer vos propres centres de données. Les deux principales approches d'utilisation d'Amazon EKS sont les suivantes :

- Norme EKS : AWS gère le plan de contrôle Kubernetes lorsque vous créez un cluster avec EKS. Les composants qui gèrent les nœuds, planifient les charges de travail, s'intègrent au AWS cloud et stockent et dimensionnent les informations du plan de contrôle pour assurer le bon fonctionnement de vos clusters sont gérés automatiquement pour vous.
- Mode automatique EKS : grâce à la fonction de mode automatique EKS, EKS étend également son contrôle à la gestion des nœuds (plan de données Kubernetes). Il simplifie la gestion de Kubernetes en provisionnant automatiquement l'infrastructure, en sélectionnant les instances de calcul optimales, en dimensionnant les ressources de manière dynamique, en optimisant continuellement les coûts, en appliquant des correctifs aux systèmes d'exploitation et en intégrant les services de sécurité. AWS

Le schéma suivant illustre la manière dont Amazon EKS intègre vos clusters Kubernetes au AWS cloud, en fonction de la méthode de création de clusters que vous choisissez :

Amazon EKS vous aide à accélérer le délai de production, à améliorer les performances, la disponibilité et la résilience, ainsi qu'à renforcer la sécurité du système. Pour plus d'informations, consultez Amazon Elastic Kubernetes Service.

Fonctions d'Amazon EKS

Amazon EKS fournit les fonctionnalités de haut niveau suivantes :

Interfaces de gestion

EKS propose plusieurs interfaces pour approvisionner, gérer et maintenir des clusters, notamment l'API Amazon EKS AWS Management Console, le CDKSDKs, la CLI AWS CloudFormation, la AWS CLI eksctl et Terraform. Pour plus d'informations, consultez Mise en route et Configuration des clusters.

Outils de contrôle d'accès

EKS s'appuie à la fois sur les fonctionnalités Kubernetes et AWS Identity and Access Management (AWS IAM) pour gérer l'accès des utilisateurs et des charges de travail. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Accès à la charge de travail à AWS ".

Ressources informatiques

En ce qui concerne les ressources de calcul, EKS propose la gamme complète de types d' EC2 instances Amazon et d' AWS innovations telles que Nitro et Graviton avec Amazon EKS pour vous permettre d'optimiser le calcul en fonction de vos charges de travail. Pour de plus amples informations, veuillez consulter Gérer le calcul.

Stockage

Le mode automatique EKS crée automatiquement des classes de stockage à l'aide de volumes EBS. À l'aide des pilotes CSI (Container Storage Interface), vous pouvez également utiliser Amazon S3, Amazon EFS, Amazon FSX et Amazon File Cache pour vos besoins de stockage d'applications. Pour de plus amples informations, veuillez consulter Stockage des données des applications.

Sécurité

Le modèle de responsabilité partagée est utilisé en ce qui concerne la sécurité dans Amazon EKS. Pour plus d'informations, consultez les meilleures pratiques en matière de sécurité, la sécurité de l'infrastructure et la sécurité Kubernetes.

Outils de surveillance

Utilisez le tableau de bord d'observabilité pour surveiller les clusters Amazon EKS. Les outils de surveillance incluent Prometheus CloudWatch, Cloudtrail et ADOT Operator. Pour plus d'informations sur les tableaux de bord, les serveurs de métriques et les autres outils, consultez les sections Coûts du cluster EKS et Kubernetes Metrics Server.

Compatibilité et support avec Kubernetes

Amazon EKS est certifié conforme à Kubernetes, ce qui vous permet de déployer des applications compatibles avec Kubernetes sans refactorisation et d'utiliser les outils et plug-ins de la communauté Kubernetes. EKS propose à la fois un support standard et eks/latest/userguide/kubernetes-versions-extended.html[extended support,type="documentation"] for Kubernetes. For more information, see eks/latest/userguide/kubernetes -versions.html [Comprendre le cycle de vie des versions de Kubernetes sur EKS, type="documentation "].

Services connexes

Services à utiliser avec Amazon EKS

Vous pouvez utiliser d'autres AWS services avec les clusters que vous déployez à l'aide d'Amazon EKS :

Amazon EC2

Bénéficiez d'une capacité de calcul évolutive à la demande avec Amazon EC2.

Amazon EBS

Associez des ressources de stockage par blocs évolutives et hautes performances avec Amazon EBS.

Amazon ECR

Stockez les images des conteneurs en toute sécurité avec Amazon ECR.

Amazon CloudWatch

Surveillez AWS les ressources et les applications en temps réel avec Amazon CloudWatch.

Amazon Prometheus

Suivez les métriques des applications conteneurisées avec Amazon Managed Service for Prometheus.

Elastic Load Balancing

Répartissez le trafic entrant sur plusieurs cibles avec Elastic Load Balancing.

Amazon GuardDuty

Détectez les menaces qui pèsent sur les clusters EKS avec Amazon GuardDuty.

AWS Centre de résilience

Évaluez la résilience du cluster EKS avec AWS Resilience Hub.

Tarification d'Amazon EKS

Amazon EKS propose une tarification par cluster basée sur la prise en charge des versions de cluster Kubernetes, la tarification pour le mode automatique Amazon EKS et la tarification par vCPU pour les nœuds hybrides Amazon EKS.

Lorsque vous utilisez Amazon EKS, vous payez séparément pour les AWS ressources que vous utilisez pour exécuter vos applications sur les nœuds de travail Kubernetes. Par exemple, si vous utilisez des nœuds de travail Kubernetes en tant qu'instances Amazon EC2 avec des volumes Amazon EBS et des IPv4 adresses publiques, la capacité de l'instance vous est facturée via Amazon EC2, la capacité du volume via Amazon EBS et l'adresse via IPv4 Amazon VPC.

Consultez les pages de tarification respectives des AWS services que vous utilisez avec vos applications Kubernetes pour obtenir des informations détaillées sur les prix.

- Pour connaître les tarifs du cluster Amazon EKS, du mode automatique Amazon EKS et des nœuds hybrides Amazon EKS, consultez les tarifs Amazon EKS.
- Pour connaître les EC2 tarifs Amazon, consultez les sections Tarification Amazon EC2 On-Demand et Amazon EC2 Spot Pricing.
- Pour les tarifs AWS Fargate, consultez AWS la section Tarification Fargate.
- Vous pouvez utiliser vos plans d'épargne pour le calcul utilisé dans les clusters Amazon EKS. Pour plus d'informations, consultez Tarification avec Savings Plans.

Cas d'utilisation courants dans Amazon EKS

Amazon EKS propose des services Kubernetes gérés robustes AWS, conçus pour optimiser les applications conteneurisées. Voici quelques-uns des cas d'utilisation les plus courants d'Amazon EKS, qui vous aideront à tirer parti de ses atouts pour répondre à vos besoins spécifiques.

Déploiement d'applications à haute disponibilité

Elastic Load Balancing vous permet de vous assurer que vos applications sont hautement disponibles sur plusieurs Zones de disponibilité.

Création d'architectures de microservices

Utilisez les fonctionnalités de découverte des services Kubernetes avec AWS Cloud Map ou Amazon VPC Lattice pour créer des systèmes résilients.

Automatisation du processus de publication des logiciels

Gérez les pipelines d'intégration et de déploiement continus (CICD) qui simplifient le processus de création, de test et de déploiement automatisés des applications.

Exécution d'applications sans serveur

Utilisez AWS Fargate avec Amazon EKS pour exécuter des applications sans serveur. Cela signifie que vous pouvez vous concentrer uniquement sur le développement d'applications, pendant qu'Amazon EKS et Fargate gèrent l'infrastructure sous-jacente.

Exécution de charges de travail liées à l'apprentissage automatique

Amazon EKS est compatible avec les cadres de machine learning les plus courants tels que TensorFlow, MXNet et PyTorch. Grâce à la prise en charge GPU, vous pouvez gérer efficacement des tâches de machine learning même complexes.

Déploiement cohérent sur site et dans le cloud

Pour simplifier l'exécution de Kubernetes dans des environnements sur site, vous pouvez utiliser les mêmes clusters, fonctionnalités et outils Amazon EKS pour exécuter des nœuds autogérés sur des AWS Outposts ou utiliser les nœuds hybrides Amazon EKS avec votre propre infrastructure. Pour les environnements autonomes et isolés, vous pouvez utiliser Amazon EKS Anywhere pour automatiser la gestion du cycle de vie des clusters Kubernetes sur votre propre infrastructure.

Exécution d'un traitement par lots rentable et de charges de travail liées aux mégadonnées

Utilisez les instances Spot pour exécuter vos traitements par lots et vos charges de travail liées au Big Data, comme Apache Hadoop et Spark, à un coût bien inférieur. Cela vous permet de profiter de la EC2 capacité Amazon inutilisée à des prix réduits.

Sécurisation des applications et garantie de la conformité

Mettez en œuvre des pratiques de sécurité strictes et maintenez la conformité avec Amazon EKS, qui s'intègre à des services de AWS sécurité tels que AWS Identity and Access Management (IAM), Amazon Virtual Private Cloud (Amazon VPC) AWS et Key Management Service AWS (KMS). Cela garantit la confidentialité et la protection des données conformément aux normes du secteur.

Architecture Amazon EKS

Amazon EKS s'aligne sur l'architecture de cluster générale de Kubernetes. Pour plus d'informations, consultez la section Composants Kubernetes dans la documentation de Kubernetes. Les sections suivantes résument certains détails supplémentaires relatifs à l'architecture d'Amazon EKS.

Plan de contrôle

Amazon EKS garantit que chaque cluster possède son propre plan de contrôle Kubernetes unique. Cette conception permet de séparer l'infrastructure de chaque cluster, sans aucun chevauchement entre les clusters ou les AWS comptes. La configuration inclut :

Composants distribués

Le plan de contrôle positionne au moins deux instances de serveur API et trois instances etcd dans trois zones de AWS disponibilité au sein d'une AWS région.

Performances optimales

Amazon EKS surveille et ajuste activement les instances du plan de contrôle afin de maintenir des performances optimales.

Résilience

Si une instance de plan de contrôle échoue, Amazon EKS la remplace rapidement, en utilisant une autre zone de disponibilité si nécessaire.

Disponibilité constante

En exécutant des clusters sur plusieurs zones de disponibilité, uncontrat de niveau de service (SLA) pour la disponibilité des points de terminaison du serveur API fiable est atteint.

Amazon EKS utilise Amazon Virtual Private Cloud (Amazon VPC) pour limiter le trafic entre les composants du plan de contrôle au sein d'un même cluster. Les composants du cluster ne peuvent pas afficher ni recevoir de communications en provenance d'autres clusters ou AWS comptes, sauf lorsqu'ils sont autorisés par les politiques de contrôle d'accès basé sur les rôles (RBAC) de Kubernetes.

Calcul

Outre le plan de contrôle, un cluster Amazon EKS possède un ensemble de machines de travail appelées nœuds. La sélection du type de nœud de cluster Amazon EKS approprié est essentielle pour répondre à vos besoins spécifiques et optimiser l'utilisation des ressources. Amazon EKS propose les types de nœuds primaires suivants :

Mode automatique EKS

Le mode automatique EKS étend la AWS gestion au-delà du plan de contrôle pour inclure le plan de données, automatisant ainsi la gestion de l'infrastructure du cluster. Il intègre les fonctionnalités principales de Kubernetes en tant que composants intégrés, notamment le dimensionnement automatique du calcul, la mise en réseau, l'équilibrage de charge, le DNS, le stockage et le support du GPU. Le mode automatique EKS gère les nœuds de manière dynamique en fonction des demandes de charge de travail, en utilisant AMIs des fonctionnalités de sécurité immuables et améliorées. Il automatise les mises à jour et les mises à niveau tout en respectant les budgets d'interruption des modules, et inclut des composants gérés qui nécessiteraient autrement une gestion complémentaire. Cette option est idéale pour les utilisateurs qui souhaitent tirer parti de l' AWS expertise day-to-day opérationnelle, minimiser les frais opérationnels et se concentrer sur le développement d'applications plutôt que sur la gestion de l'infrastructure.

AWS Fargate

Fargateest un moteur de calcul sans serveur pour les conteneurs qui élimine le besoin de gérer les instances sous-jacentes. Avec Fargate, vous spécifiez les besoins en ressources de votre application et provisionnez, AWS dimensionnez et entretenez automatiquement l'infrastructure. Cette option est idéale pour les utilisateurs qui privilégient ease-of-use et souhaitent se concentrer sur le développement et le déploiement d'applications plutôt que sur la gestion de l'infrastructure.

Charpentier

Karpenter est un autoscaler de cluster Kubernetes flexible et performant qui permet d'améliorer la disponibilité des applications et l'efficacité des clusters. Karpenter lance des ressources de calcul bien dimensionnées en réponse à l'évolution de la charge des applications. Cette option permet de fournir des ressources de just-in-time calcul répondant aux exigences de votre charge de travail.

Groupes de nœuds gérés

Les groupes de nœuds gérés sont un mélange d'automatisation et de personnalisation permettant de gérer un ensemble d' EC2 instances Amazon au sein d'un cluster Amazon EKS. AWS prend en charge des tâches telles que l'application de correctifs, la mise à jour et le dimensionnement des nœuds, simplifiant ainsi les aspects opérationnels. En parallèle, les arguments `kubelet` personnalisés sont pris en charge, ce qui ouvre la voie à des politiques avancées de gestion du processeur et de la mémoire. En outre, ils renforcent la sécurité grâce aux rôles AWS Identity and Access Management (IAM) pour les comptes de service, tout en réduisant le besoin d'autorisations distinctes par cluster.

Nœuds autogérés

Les nœuds autogérés offrent un contrôle total sur vos EC2 instances Amazon au sein d'un cluster Amazon EKS. Vous êtes chargé de gérer, de mettre à l'échelle et de maintenir les nœuds, ce qui vous donne un contrôle total sur l'infrastructure sous-jacente. Cette option convient aux utilisateurs qui ont besoin d'un contrôle granulaire et d'une personnalisation de leurs nœuds et qui sont prêts à investir du temps dans la gestion et la maintenance de leur infrastructure.

Nœuds hybrides Amazon EKS

Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et périphérique en tant que nœuds dans des clusters Amazon EKS. Amazon EKS Hybrid Nodes unifie la gestion de Kubernetes dans tous les environnements et délègue la gestion du plan de contrôle Kubernetes à vos applications sur site et de périphérie. AWS

Concepts de Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) AWS est un service géré basé sur le projet open source Kubernetes. Bien que vous deviez savoir certaines choses sur la manière dont le service Amazon EKS s'intègre au AWS cloud (en particulier lorsque vous créez un cluster Amazon EKS pour la première fois), une fois qu'il est opérationnel, vous utilisez votre cluster Amazon EKS de la même manière que n'importe quel autre cluster Kubernetes. Ainsi, pour commencer à gérer des clusters Kubernetes et à déployer des charges de travail, vous devez au moins avoir une connaissance de base des concepts de Kubernetes.

Cette page divise les concepts de Kubernetes en trois sections :the section called “Pourquoi Kubernetes ?”,, etthe section called “Clusters”. the section called “Charges de travail” La première section décrit l'intérêt de l'exécution d'un service Kubernetes, en particulier en tant que service géré tel qu'Amazon EKS. La section Charges de travail explique comment les applications Kubernetes sont créées, stockées, exécutées et gérées. La section Clusters présente les différents composants qui constituent les clusters Kubernetes et quelles sont vos responsabilités en matière de création et de maintenance de clusters Kubernetes.

Rubriques

- Pourquoi Kubernetes ?
- Clusters
- Charges de travail
- Étapes suivantes

Au fur et à mesure que vous parcourez ce contenu, des liens vous redirigeront vers des descriptions supplémentaires des concepts Kubernetes dans la documentation Amazon EKS et Kubernetes, au cas où vous souhaiteriez approfondir l'un des sujets abordés ici. Pour en savoir plus sur la manière dont Amazon EKS implémente le plan de contrôle et les fonctionnalités de calcul de Kubernetes, consultez. the section called "Architecture"

Pourquoi Kubernetes ?

Kubernetes a été conçu pour améliorer la disponibilité et l'évolutivité lors de l'exécution d'applications conteneurisées critiques de qualité pour la production. Plutôt que de simplement exécuter Kubernetes sur une seule machine (bien que cela soit possible), Kubernetes atteint ces objectifs en vous permettant d'exécuter des applications sur des ensembles d'ordinateurs qui peuvent s'étendre ou se contracter pour répondre à la demande. Kubernetes inclut des fonctionnalités qui vous permettent de :

- Déployer des applications sur plusieurs machines (à l'aide de conteneurs déployés dans des pods)
- Surveillez l'état des conteneurs et redémarrez les conteneurs défaillants
- Agrandir ou diminuer les conteneurs en fonction de la charge
- Mettre à jour les conteneurs avec les nouvelles versions
- Allouer des ressources entre les conteneurs
- Équilibrez le trafic entre les machines

Le fait que Kubernetes automatise ces types de tâches complexes permet aux développeurs d'applications de se concentrer sur la création et l'amélioration de leurs charges de travail applicatives, plutôt que de se soucier de l'infrastructure. Le développeur crée généralement des fichiers de configuration, au format YAML, qui décrivent l'état souhaité de l'application. Cela peut inclure les conteneurs à exécuter, les limites de ressources, le nombre de répliques de Pod, l'allocation CPU/mémoire, les règles d'affinité, etc.

Attributs de Kubernetes

Pour atteindre ses objectifs, Kubernetes possède les caractéristiques suivantes :

- Conteneurisé — Kubernetes est un outil d'orchestration de conteneurs. Pour utiliser Kubernetes, vous devez d'abord conteneuriser vos applications. Selon le type d'application, il peut s'agir d'un ensemble de microservices, de tâches par lots ou d'autres formes. Vos applications peuvent ensuite tirer parti d'un flux de travail Kubernetes qui englobe un vaste écosystème d'outils, dans

lequel les conteneurs peuvent être stockés sous forme d'images dans un registre de conteneurs, déployés sur un cluster Kubernetes et exécutés sur un nœud disponible. Vous pouvez créer et tester des conteneurs individuels sur votre ordinateur local avec Docker ou un autre environnement d'exécution de conteneurs, avant de les déployer sur votre cluster Kubernetes.

- Évolutif : si la demande pour vos applications dépasse la capacité des instances en cours d'exécution de ces applications, Kubernetes est en mesure de monter en gamme. Selon les besoins, Kubernetes peut déterminer si les applications ont besoin de plus de processeur ou de mémoire et réagir en augmentant automatiquement la capacité disponible ou en utilisant une plus grande partie de la capacité existante. Le dimensionnement peut être effectué au niveau du pod, s'il y a suffisamment de calcul disponible pour exécuter un plus grand nombre d'instances de l'application (scalage automatique du pod horizontal), ou au niveau du nœud, si davantage de nœuds doivent être créés pour gérer l'augmentation de capacité (Cluster Autoscaler ou Karpenter). Comme la capacité n'est plus nécessaire, ces services peuvent supprimer les pods inutiles et arrêter les nœuds inutiles.
- Disponible : si une application ou un nœud devient défaillant ou indisponible, Kubernetes peut déplacer les charges de travail en cours vers un autre nœud disponible. Vous pouvez résoudre le problème en supprimant simplement une instance active d'une charge de travail ou d'un nœud qui exécute vos charges de travail. En fin de compte, les charges de travail peuvent être transférées sur d'autres sites si elles ne peuvent plus fonctionner là où elles se trouvent.
- Déclaratif — Kubernetes utilise la réconciliation active pour vérifier en permanence que l'état que vous déclarez pour votre cluster correspond à l'état réel. En appliquant des objets Kubernetes à un cluster, généralement par le biais de fichiers de configuration au format YAML, vous pouvez, par exemple, demander de démarrer les charges de travail que vous souhaitez exécuter sur votre cluster. Vous pouvez ultérieurement modifier les configurations pour utiliser une version ultérieure d'un conteneur ou allouer plus de mémoire, par exemple. Kubernetes fera ce qu'il faut pour établir l'état souhaité. Cela peut inclure l'augmentation ou la désactivation des nœuds, l'arrêt et le redémarrage des charges de travail ou le retrait de conteneurs mis à jour.
- Composable : étant donné qu'une application est généralement composée de plusieurs composants, vous souhaitez être en mesure de gérer ensemble un ensemble de ces composants (souvent représentés par plusieurs conteneurs). Docker Compose permet de le faire directement avec Docker, mais la commande Kubernetes Kompose peut vous aider à le faire avec Kubernetes. Voir Translate a Docker Compose File to Kubernetes Resources pour un exemple de procédure à suivre.
- Extensible — Contrairement aux logiciels propriétaires, le projet open source Kubernetes est conçu pour vous permettre d'étendre Kubernetes comme bon vous semble pour répondre à vos

besoins. APIs et les fichiers de configuration sont ouverts aux modifications directes. Les tiers sont invités à créer leurs propres contrôleurs, afin d'étendre à la fois l'infrastructure et les fonctionnalités Kubernetes pour les utilisateurs finaux. Les webhooks vous permettent de configurer des règles de cluster pour appliquer les politiques et vous adapter à l'évolution des conditions. Pour plus d'idées sur la manière d'étendre les clusters Kubernetes, consultez la section Extension de Kubernetes.

- Portable — De nombreuses entreprises ont standardisé leurs opérations sur Kubernetes, car cela leur permet de gérer tous leurs besoins en matière d'applications de la même manière. Les développeurs peuvent utiliser les mêmes pipelines pour créer et stocker des applications conteneurisées. Ces applications peuvent ensuite être déployées sur des clusters Kubernetes exécutés sur site, dans le cloud, sur des point-of-sales terminaux de restaurants ou sur des appareils IoT dispersés sur les sites distants de l'entreprise. Sa nature open source permet aux utilisateurs de développer ces distributions spéciales de Kubernetes, ainsi que les outils nécessaires pour les gérer.

Gestion de Kubernetes

Le code source de Kubernetes est disponible gratuitement. Ainsi, avec votre propre équipement, vous pouvez installer et gérer Kubernetes vous-même. Cependant, l'autogestion de Kubernetes nécessite une expertise opérationnelle approfondie et sa maintenance demande du temps et des efforts. Pour ces raisons, la plupart des personnes qui déploient des charges de travail de production choisissent un fournisseur cloud (tel qu'Amazon EKS) ou un fournisseur sur site (tel qu'Amazon EKS Anywhere) avec sa propre distribution Kubernetes testée et le support d'experts Kubernetes. Cela vous permet de vous décharger d'une grande partie des tâches lourdes indifférenciées nécessaires à la maintenance de vos clusters, notamment :

- Matériel : si vous ne disposez pas de matériel pour exécuter Kubernetes selon vos besoins, un fournisseur de cloud tel qu' AWS Amazon EKS peut vous faire économiser sur les coûts initiaux. Avec Amazon EKS, cela signifie que vous pouvez utiliser les meilleures ressources cloud proposées AWS, notamment les instances informatiques (Amazon Elastic Compute Cloud), votre propre environnement privé (Amazon VPC), la gestion centralisée des identités et des autorisations (IAM) et le stockage (Amazon EBS). AWS gère les ordinateurs, les réseaux, les centres de données et tous les autres composants physiques nécessaires au fonctionnement de Kubernetes. De même, vous n'avez pas à planifier votre centre de données de manière à gérer la capacité maximale les jours où la demande est la plus forte. Pour Amazon EKS Anywhere ou les autres clusters Kubernetes locaux, vous êtes responsable de la gestion de l'infrastructure utilisée dans vos déploiements Kubernetes, mais vous pouvez toujours compter sur vous pour vous aider AWS à maintenir Kubernetes à jour.

- Gestion du plan de contrôle : Amazon EKS gère la sécurité et la disponibilité du plan de contrôle Kubernetes AWS hébergé, qui est chargé de planifier les conteneurs, de gérer la disponibilité des applications et d'autres tâches clés, afin que vous puissiez vous concentrer sur les charges de travail de vos applications. Si votre cluster tombe en panne, vous AWS devriez avoir les moyens de rétablir son état de fonctionnement. Pour Amazon EKS Anywhere, vous devez gérer vous-même le plan de contrôle.
- Mises à niveau testées : lorsque vous mettez à niveau vos clusters, vous pouvez compter sur Amazon EKS ou Amazon EKS Anywhere pour fournir des versions testées de leurs distributions Kubernetes.
- Extensions : il existe des centaines de projets conçus pour étendre Kubernetes et fonctionner avec Kubernetes. Vous pouvez les ajouter à l'infrastructure de votre cluster ou les utiliser pour faciliter l'exécution de vos charges de travail. Au lieu de créer et de gérer vous-même ces modules complémentaires AWS , the section called "Modules complémentaires Amazon EKS" fournissez que vous pouvez utiliser avec vos clusters. Amazon EKS Anywhere propose des packages sélectionnés qui incluent des versions de nombreux projets open source populaires. Vous n'avez donc pas à créer vous-même le logiciel ni à gérer les correctifs de sécurité critiques, les corrections de bogues ou les mises à niveau. De même, si les valeurs par défaut répondent à vos besoins, il est courant que très peu de configuration de ces modules complémentaires soient nécessaires. Consultez the section called "Étendre les clusters" pour plus de détails sur l'extension de votre cluster avec des modules complémentaires.

Kubernetes en action

Le schéma suivant montre les principales activités que vous devez effectuer en tant qu'administrateur Kubernetes ou développeur d'applications pour créer et utiliser un cluster Kubernetes. Ce faisant, il illustre la manière dont les composants Kubernetes interagissent les uns avec les autres, en utilisant le AWS cloud comme exemple du fournisseur de cloud sous-jacent.

Un administrateur Kubernetes crée le cluster Kubernetes à l'aide d'un outil spécifique au type de fournisseur sur lequel le cluster sera construit. Cet exemple utilise le AWS cloud comme fournisseur, qui propose le service Kubernetes géré appelé Amazon EKS. Le service géré alloue automatiquement les ressources nécessaires à la création du cluster, notamment en créant deux nouveaux clouds privés virtuels VPCs (Amazon) pour le cluster, en configurant le réseau et en mappant les autorisations Kubernetes directement dans le nouveau VPCs cloud de gestion des actifs. Le service géré veille également à ce que les services du plan de contrôle disposent d'emplacements où s'exécuter et n'alloue aucune EC2 instance Amazon ou plus en tant que nœuds Kubernetes pour exécuter les charges de travail. AWS gère lui-même un Amazon VPC pour le plan de contrôle, tandis que l'autre Amazon VPC contient les nœuds clients qui exécutent les charges de travail.

À l'avenir, de nombreuses tâches de l'administrateur Kubernetes sont effectuées à l'aide d'outils Kubernetes tels que. `kubectl` Cet outil envoie les demandes de services directement au plan de contrôle du cluster. La manière dont les requêtes et les modifications sont apportées au cluster est alors très similaire à celle que vous utiliseriez sur n'importe quel cluster Kubernetes.

Un développeur d'applications souhaitant déployer des charges de travail sur ce cluster peut effectuer plusieurs tâches. Le développeur doit intégrer l'application dans une ou plusieurs images

de conteneur, puis transférer ces images vers un registre de conteneurs accessible au cluster Kubernetes. AWS propose l'Amazon Elastic Container Registry (Amazon ECR) à cette fin.

Pour exécuter l'application, le développeur peut créer des fichiers de configuration au format YAML qui indiquent au cluster comment exécuter l'application, notamment les conteneurs à extraire du registre et comment les encapsuler dans des pods. Le plan de contrôle (planificateur) planifie les conteneurs sur un ou plusieurs nœuds et le moteur d'exécution des conteneurs sur chaque nœud extrait et exécute les conteneurs nécessaires. Le développeur peut également configurer un équilibreur de charge d'application pour équilibrer le trafic vers les conteneurs disponibles exécutés sur chaque nœud et exposer l'application afin qu'elle soit disponible sur un réseau public au monde extérieur. Une fois tout cela fait, une personne souhaitant utiliser l'application peut se connecter au point de terminaison de l'application pour y accéder.

Les sections suivantes présentent en détail chacune de ces fonctionnalités, du point de vue des clusters et des charges de travail Kubernetes.

Clusters

Si votre travail consiste à démarrer et à gérer des clusters Kubernetes, vous devez savoir comment les clusters Kubernetes sont créés, améliorés, gérés et supprimés. Vous devez également savoir quels sont les composants qui constituent un cluster et ce que vous devez faire pour les maintenir à jour.

Les outils de gestion des clusters gèrent le chevauchement entre les services Kubernetes et le fournisseur de matériel sous-jacent. C'est pourquoi l'automatisation de ces tâches est généralement effectuée par le fournisseur Kubernetes (tel qu'Amazon EKS ou Amazon EKS Anywhere) à l'aide d'outils spécifiques au fournisseur. Par exemple, pour démarrer un cluster Amazon EKS, vous pouvez l'utiliser`eksctl create cluster`, tandis que pour Amazon EKS Anywhere, vous pouvez utiliser`eksctl anywhere create cluster`. Notez que bien que ces commandes créent un cluster Kubernetes, elles sont spécifiques au fournisseur et ne font pas partie du projet Kubernetes lui-même.

Outils de création et de gestion de clusters

Le projet Kubernetes propose des outils permettant de créer manuellement un cluster Kubernetes. Ainsi, si vous souhaitez installer Kubernetes sur une seule machine, ou exécuter le plan de contrôle sur une machine et ajouter des nœuds manuellement, vous pouvez utiliser des outils CLI tels que kind, minikube ou kubeadm répertoriés sous Outils d'installation Kubernetes. Pour simplifier et

automatiser le cycle de vie complet de la création et de la gestion des clusters, il est beaucoup plus facile d'utiliser des outils pris en charge par un fournisseur Kubernetes établi, tel qu'Amazon EKS ou Amazon EKS Anywhere.

Dans AWS le cloud, vous pouvez créer des clusters Amazon EKS à l'aide d'outils CLI, tels que eksctl, ou d'outils plus déclaratifs, tels que Terraform (voir Amazon EKS Blueprints for Terraform). Vous pouvez également créer un cluster à partir du AWS Management Console. Consultez les fonctionnalités d'Amazon EKS pour obtenir une liste de ce que vous pouvez obtenir avec Amazon EKS. Les responsabilités de Kubernetes qu'Amazon EKS assume pour vous incluent :

- Plan de contrôle géré :AWS garantit que le cluster Amazon EKS est disponible et évolutif, car il gère le plan de contrôle pour vous et le rend disponible dans toutes les zones de AWS disponibilité.
- Gestion des nœuds : au lieu d'ajouter manuellement des nœuds, vous pouvez demander à Amazon EKS de créer des nœuds automatiquement selon vos besoins, à l'aide de Managed Node Groups (voirthe section called "Groupes de nœuds gérés") ou de Karpenter. Les groupes de nœuds gérés sont intégrés à Kubernetes Cluster Autoscaling. À l'aide des outils de gestion des nœuds, vous pouvez réaliser des économies, notamment grâce aux instances Spot, à la consolidation et à la disponibilité des nœuds, en utilisant les fonctionnalités de planification pour définir la manière dont les charges de travail sont déployées et les nœuds sélectionnés.
- Mise en réseau de clusters : à l'aide de CloudFormation modèles, `eksctl` configure la mise en réseau entre les composants du plan de contrôle et du plan de données (nœud) dans le cluster Kubernetes. Il met également en place des points de terminaison par lesquels les communications internes et externes peuvent avoir lieu. Consultez Démystifier le réseau de clusters pour les nœuds de travail Amazon EKS pour plus de détails. Les communications entre les pods dans Amazon EKS sont effectuées à l'aide d'Amazon EKS Pod Identities (voirthe section called "Identité du pod"), qui permet aux Pods de tirer parti des méthodes AWS cloud de gestion des informations d'identification et des autorisations.
- Modules complémentaires : Amazon EKS vous évite d'avoir à créer et à ajouter des composants logiciels couramment utilisés pour prendre en charge les clusters Kubernetes. Par exemple, lorsque vous créez un cluster Amazon EKS à partir du AWS Management Console, il ajoute automatiquement les modules complémentaires Amazon EKS kube-proxy (the section called "kube-proxy"), Amazon VPC CNI pour Kubernetes (the section called "CNI Amazon VPC") et CoreDNS (). the section called "CoreDNS" Consultez the section called "Modules complémentaires Amazon EKS" pour en savoir plus sur ces modules complémentaires, y compris la liste des modules disponibles.

Pour exécuter vos clusters sur vos propres ordinateurs et réseaux locaux, Amazon propose Amazon EKS Anywhere. Au lieu que le AWS Cloud soit le fournisseur, vous avez le choix d'exécuter Amazon EKS Anywhere sur les plateformes VMWare vSphere, bare metal (fournisseur Tinkerbell) CloudStack, Snow ou Nutanix en utilisant votre propre équipement.

Amazon EKS Anywhere est basé sur le même logiciel Amazon EKS Distro que celui utilisé par Amazon EKS. Cependant, Amazon EKS Anywhere s'appuie sur différentes implémentations de l'interface CAPI (Kubernetes Cluster API) pour gérer le cycle de vie complet des machines d'un cluster Amazon EKS Anywhere (comme CAPV pour vSphere et CAPC pour). CloudStack Comme l'ensemble du cluster fonctionne sur votre équipement, vous assumez la responsabilité supplémentaire de gérer le plan de contrôle et de sauvegarder ses données (voir `etcd` plus loin dans ce document).

Composants du cluster

Les composants du cluster Kubernetes sont divisés en deux zones principales : le plan de contrôle et les nœuds de travail. Les composants du plan de contrôle gèrent le cluster et fournissent un accès à celui-ci APIs. Les nœuds de travail (parfois simplement appelés nœuds) fournissent les emplacements où les charges de travail réelles sont exécutées. Les composants du nœud sont des services qui s'exécutent sur chaque nœud pour communiquer avec le plan de contrôle et exécuter des conteneurs. L'ensemble des nœuds de travail de votre cluster est appelé plan de données.

Plan de contrôle

Le plan de contrôle est constitué d'un ensemble de services qui gèrent le cluster. Ces services peuvent tous être exécutés sur un seul ordinateur ou peuvent être répartis sur plusieurs ordinateurs. En interne, elles sont appelées instances du plan de contrôle (CPIs). Le mode CPIs d'exécution dépend de la taille du cluster et des exigences en matière de haute disponibilité. À mesure que la demande augmente dans le cluster, un service de plan de contrôle peut évoluer pour fournir davantage d'instances de ce service, les demandes étant équilibrées entre les instances.

Les tâches effectuées par les composants du plan de contrôle Kubernetes sont les suivantes :

- Communication avec les composants du cluster (serveur d'API) — Le serveur d'API (kube-apiserver) expose l'API Kubernetes afin que les demandes au cluster puissent être effectuées à la fois depuis l'intérieur et l'extérieur du cluster. En d'autres termes, les demandes d'ajout ou de modification des objets d'un cluster (pods, services, nœuds, etc.) peuvent provenir de commandes externes, telles que des demandes `kubectl` d'exécution d'un pod. De même, des demandes

peuvent être adressées par le serveur API aux composants du cluster, par exemple une requête au `kubelet` service concernant l'état d'un Pod.

- Stocker les données relatives au cluster (magasin de valeurs **`etcd`** clés) : le `etcd` service joue un rôle essentiel dans le suivi de l'état actuel du cluster. Si le `etcd` service devenait inaccessible, vous ne pourriez pas mettre à jour ou demander l'état du cluster, même si les charges de travail continueraient à s'exécuter pendant un certain temps. Pour cette raison, les clusters critiques disposent généralement de plusieurs instances équilibrées du `etcd` service exécutées simultanément et effectuent des sauvegardes périodiques du magasin de valeurs `etcd` clés en cas de perte ou de corruption de données. N'oubliez pas que, dans Amazon EKS, tout cela est géré automatiquement par défaut. Amazon EKS Anywhere fournit des instructions pour la sauvegarde et la restauration etcd. Consultez le modèle de données etcd pour savoir comment `etcd` gérer les données.
- Planifier des pods vers des nœuds (planificateur) : les demandes de démarrage ou d'arrêt d'un pod dans Kubernetes sont dirigées vers le planificateur Kubernetes (kube-scheduler). Comme un cluster peut comporter plusieurs nœuds capables d'exécuter le Pod, il appartient au planificateur de choisir sur quel nœud (ou quels nœuds, dans le cas de répliques) le Pod doit s'exécuter. Si la capacité disponible est insuffisante pour exécuter le Pod demandé sur un nœud existant, la demande échouera, sauf si vous avez pris d'autres dispositions. Ces dispositions pourraient inclure des services d'activation tels que Managed Node Groups (the section called "Groupes de nœuds gérés") ou Karpenter, capables de démarrer automatiquement de nouveaux nœuds pour gérer les charges de travail.
- Maintenir les composants dans l'état souhaité (Controller Manager) — Le Kubernetes Controller Manager s'exécute comme un processus daemon (kube-controller-manager) pour surveiller l'état du cluster et apporter des modifications au cluster afin de rétablir les états attendus. En particulier, plusieurs contrôleurs surveillent différents objets Kubernetes, notamment a,, `statefulset-controller` `endpoint-controller` `cronjob-controller``node-controller`, et d'autres.
- Gérer les ressources du cloud (Cloud Controller Manager) : les interactions entre Kubernetes et le fournisseur de cloud qui exécute les demandes relatives aux ressources du centre de données sous-jacent sont gérées par le Cloud Controller Manager (). cloud-controller-manager Les contrôleurs gérés par le Cloud Controller Manager peuvent inclure un contrôleur de route (pour configurer les routes du réseau cloud), un contrôleur de service (pour utiliser les services d'équilibrage de charge dans le cloud) et un contrôleur du cycle de vie des nœuds (pour maintenir les nœuds synchronisés avec Kubernetes tout au long de leur cycle de vie).

Nœuds de travail (plan de données)

Pour un cluster Kubernetes à nœud unique, les charges de travail s'exécutent sur la même machine que le plan de contrôle. Cependant, une configuration plus standard consiste à avoir un ou plusieurs systèmes informatiques distincts (nœuds) dédiés à l'exécution des charges de travail Kubernetes.

Lorsque vous créez un cluster Kubernetes pour la première fois, certains outils de création de cluster vous permettent de configurer un certain nombre de nœuds à ajouter au cluster (soit en identifiant les systèmes informatiques existants, soit en demandant au fournisseur d'en créer de nouveaux). Avant d'ajouter des charges de travail à ces systèmes, des services sont ajoutés à chaque nœud pour implémenter les fonctionnalités suivantes :

- Gérer chaque nœud (**`kubelet`**) — Le serveur API communique avec le service kubelet exécuté sur chaque nœud pour s'assurer que le nœud est correctement enregistré et que les pods demandés par le planificateur fonctionnent. Le kubelet peut lire les manifestes des pods et configurer des volumes de stockage ou d'autres fonctionnalités nécessaires aux pods sur le système local. Il peut également vérifier l'état des conteneurs gérés localement.
- Exécuter des conteneurs sur un nœud (runtime de conteneur) : le Container Runtime de chaque nœud gère les conteneurs demandés pour chaque pod attribué au nœud. Cela signifie qu'il peut extraire les images du conteneur du registre approprié, exécuter le conteneur, l'arrêter et répondre aux requêtes concernant le conteneur. Le runtime du conteneur par défaut est containerd. À partir de Kubernetes 1.24, l'intégration spéciale de Docker (`dockershim`) qui pouvait être utilisée comme environnement d'exécution du conteneur a été supprimée de Kubernetes. Bien que vous puissiez toujours utiliser Docker pour tester et exécuter des conteneurs sur votre système local, pour utiliser Docker avec Kubernetes, vous devez désormais installer Docker Engine sur chaque nœud pour l'utiliser avec Kubernetes.
- Gérer le réseau entre les conteneurs (**`kube-proxy`**) — Pour pouvoir prendre en charge la communication entre les pods, Kubernetes utilise une fonctionnalité appelée service pour configurer des réseaux de pods qui suivent les adresses IP et les ports associés à ces pods. Le service kube-proxy s'exécute sur chaque nœud pour permettre la communication entre les pods.

Étendre les clusters

Vous pouvez ajouter certains services à Kubernetes pour prendre en charge le cluster, mais ils ne sont pas exécutés dans le plan de contrôle. Ces services s'exécutent souvent directement sur les nœuds de l'espace de noms du système Kube ou dans son propre espace de noms (comme c'est souvent le cas avec les fournisseurs de services tiers). Le service CoreDNS, qui fournit des services

DNS au cluster, en est un exemple courant. Reportez-vous à la section Découverte des services intégrés pour savoir comment voir quels services de cluster s'exécutent dans Kube-System sur votre cluster.

Il existe différents types de modules complémentaires que vous pouvez envisager d'ajouter à vos clusters. Pour préserver la santé de vos clusters, vous pouvez ajouter des fonctionnalités d'observabilité (voirSurveiller les clusters) qui vous permettent d'effectuer des tâches telles que la journalisation, l'audit et les métriques. Grâce à ces informations, vous pouvez résoudre les problèmes qui surviennent, souvent par le biais des mêmes interfaces d'observabilité. Des exemples de ces types de services incluent Amazon CloudWatch (voirthe section called "Amazon CloudWatch") GuardDuty, AWS Distro pour OpenTelemetry, le plugin Amazon VPC CNI pour Kubernetes (voirthe section called "CNI Amazon VPC") et Grafana Kubernetes Monitoring. Pour le stockage (voirStockage des données des applications), les extensions d'Amazon EKS incluent le pilote Amazon Elastic Block Store CSI (voirthe section called "Amazon EBS"), le pilote Amazon Elastic File System CSI (voirthe section called "Amazon EFS") et plusieurs modules complémentaires de stockage tiers (tels que le pilote Amazon FSx for NetApp ONTAP CSI). the section called "Amazon FSx pour NetApp ONTAP"

Pour une liste plus complète des modules complémentaires Amazon EKS disponibles, consultezthe section called "Modules complémentaires Amazon EKS".

Charges de travail

Kubernetes définit un workload comme « une application exécutée sur Kubernetes ». Cette application peut consister en un ensemble de microservices exécutés sous forme de conteneurs dans des pods, ou peut être exécutée sous forme de traitement par lots ou d'autres types d'applications. Le travail de Kubernetes est de s'assurer que les demandes que vous faites pour configurer ou déployer ces objets sont exécutées. En tant que personne déployant des applications, vous devez apprendre comment les conteneurs sont créés, comment les pods sont définis et quelles méthodes vous pouvez utiliser pour les déployer.

Conteneurs

L'élément le plus élémentaire d'une charge de travail d'application que vous déployez et gérez dans Kubernetes est un pod. Un Pod représente un moyen de contenir les composants d'une application et de définir des spécifications décrivant les attributs du Pod. Comparez cela à quelque chose comme un package RPM ou Deb, qui regroupe des logiciels pour un système Linux, mais ne s'exécute pas lui-même en tant qu'entité.

Le Pod étant la plus petite unité déployable, il contient généralement un seul conteneur. Cependant, plusieurs conteneurs peuvent se trouver dans un Pod dans les cas où les conteneurs sont étroitement couplés. Par exemple, un conteneur de serveur Web peut être empaqueté dans un pod avec un conteneur de type sidecar qui peut fournir des services de journalisation, de surveillance ou tout autre service étroitement lié au conteneur de serveur Web. Dans ce cas, le fait d'être dans le même Pod garantit que pour chaque instance en cours d'exécution du Pod, les deux conteneurs s'exécutent toujours sur le même nœud. De même, tous les conteneurs d'un Pod partagent le même environnement, les conteneurs d'un Pod s'exécutant comme s'ils se trouvaient sur le même hôte isolé. Cela a pour effet que les conteneurs partagent une adresse IP unique qui donne accès au Pod et que les conteneurs peuvent communiquer entre eux comme s'ils s'exécutaient sur leur propre hôte local.

Les spécifications du pod (PodSpec) définissent l'état souhaité du pod. Vous pouvez déployer un pod individuel ou plusieurs pods en utilisant les ressources de charge de travail pour gérer les modèles de pods. Les ressources de charge de travail incluent les déploiements (pour gérer plusieurs répliques de pods), StatefulSets(pour déployer des pods qui doivent être uniques, tels que les pods de base de données) et DaemonSets(où un pod doit fonctionner en continu sur chaque nœud). Plus d'informations à ce sujet plus tard.

Alors qu'un pod est la plus petite unité que vous déployez, un conteneur est la plus petite unité que vous créez et gérez.

Conteneurs de construction

Le Pod n'est en réalité qu'une structure autour d'un ou de plusieurs conteneurs, chaque conteneur contenant lui-même le système de fichiers, les exécutables, les fichiers de configuration, les bibliothèques et les autres composants nécessaires à l'exécution de l'application. Parce qu'une société appelée Docker Inc. a d'abord popularisé les conteneurs, certaines personnes appellent les conteneurs Docker Containers. Cependant, l'Open Container Initiative a depuis défini les durées d'exécution des conteneurs, les images et les méthodes de distribution pour le secteur. Ajoutez à cela le fait que les conteneurs ont été créés à partir de nombreuses fonctionnalités Linux existantes, d'autres les appellent souvent conteneurs OCI, conteneurs Linux ou simplement conteneurs.

Lorsque vous créez un conteneur, vous commencez généralement par un Dockerfile (littéralement nommé ainsi). Dans ce Dockerfile, vous identifiez :

- Une image de base : une image de conteneur de base est un conteneur généralement créé à partir d'une version minimale du système de fichiers d'un système d'exploitation (tel que Red Hat

Enterprise Linux ou Ubuntu) ou d'un système minimal amélioré pour fournir des logiciels permettant d'exécuter des types d'applications spécifiques (tels que des applications nodejs ou python).

- Logiciel d'application — Vous pouvez ajouter votre logiciel d'application à votre conteneur de la même manière que vous l'ajouteriez à un système Linux. Par exemple, dans votre Dockerfile, vous pouvez exécuter `npm` et `yarn` installer une application Java ou `dnf` installer `yum` des packages RPM. En d'autres termes, à l'aide d'une commande RUN dans un Dockerfile, vous pouvez exécuter n'importe quelle commande disponible dans le système de fichiers de votre image de base pour installer un logiciel ou configurer un logiciel à l'intérieur de l'image de conteneur obtenue.
- Instructions — La référence Dockerfile décrit les instructions que vous pouvez ajouter à un Dockerfile lorsque vous le configurez. Il s'agit notamment des instructions utilisées pour créer le contenu du conteneur lui-même (`ADD`ou des `COPY` fichiers du système local), identifier les commandes à exécuter lorsque le conteneur est exécuté (`CMD`ou`ENTRYPOINT`) et connecter le conteneur au système sur lequel il s'exécute (en identifiant le `USER` conteneur sous lequel exécuter, un local `VOLUME` à monter ou les ports vers lesquels`EXPOSE`).

Bien que la `docker` commande et le service soient traditionnellement utilisés pour créer des conteneurs (`docker build`), d'autres outils sont disponibles pour créer des images de conteneurs, notamment podman et nerdctl. Consultez la section Création de meilleures images de conteneurs ou Présentation de Docker Build pour en savoir plus sur la création de conteneurs.

Conteneurs de stockage

Une fois que vous avez créé votre image de conteneur, vous pouvez la stocker dans un registre de distribution de conteneurs sur votre poste de travail ou dans un registre de conteneurs public. La gestion d'un registre de conteneurs privé sur votre poste de travail vous permet de stocker des images de conteneurs localement, les rendant ainsi facilement accessibles.

Pour stocker des images de conteneurs de manière plus publique, vous pouvez les transférer vers un registre de conteneurs public. Les registres publics de conteneurs fournissent un emplacement central pour le stockage et la distribution des images des conteneurs. Les registres de conteneurs publics incluent notamment le registre Amazon Elastic Container Registry, le registre Red Hat Quay et le registre Docker Hub.

Lorsque vous exécutez des charges de travail conteneurisées sur Amazon Elastic Kubernetes Service (Amazon EKS), nous vous recommandons de récupérer des copies des images officielles Docker stockées dans Amazon Elastic Container Registry. Amazon ECR stocke ces images depuis 2021. Vous pouvez rechercher des images de conteneurs populaires dans la galerie publique

Amazon ECR, et plus particulièrement pour les images Docker Hub, vous pouvez effectuer une recherche dans la galerie Amazon ECR Docker.

Conteneurs de course

Les conteneurs étant conçus dans un format standard, un conteneur peut s'exécuter sur n'importe quelle machine capable d'exécuter un environnement d'exécution de conteneur (comme Docker) et dont le contenu correspond à l'architecture de la machine locale (telle que `x86_64` ou`arm`). Pour tester un conteneur ou simplement l'exécuter sur votre bureau local, vous pouvez utiliser `podman run` les commandes `docker run` or pour démarrer un conteneur sur l'hôte local. Pour Kubernetes, cependant, chaque nœud de travail dispose d'un environnement d'exécution de conteneur déployé et c'est à Kubernetes de demander à un nœud d'exécuter un conteneur.

Une fois qu'un conteneur a été assigné pour s'exécuter sur un nœud, le nœud vérifie si la version demandée de l'image du conteneur existe déjà sur le nœud. Si ce n'est pas le cas, Kubernetes demande au moteur d'exécution du conteneur d'extraire ce conteneur du registre de conteneurs approprié, puis de l'exécuter localement. N'oubliez pas qu'une image de conteneur fait référence au package logiciel qui est déplacé entre votre ordinateur portable, le registre des conteneurs et les nœuds Kubernetes. Un conteneur fait référence à une instance en cours d'exécution de cette image.

Capsules

Une fois que vos conteneurs sont prêts, l'utilisation des pods inclut la configuration, le déploiement et la mise à disposition des pods.

Configuration des pods

Lorsque vous définissez un Pod, vous lui attribuez un ensemble d'attributs. Ces attributs doivent inclure au moins le nom du pod et l'image du conteneur à exécuter. Cependant, vous souhaitez également configurer de nombreux autres éléments avec les définitions de votre pod (consultez la PodSpecpage pour plus de détails sur ce qui peut être inclus dans un pod). Il s'agit des licences suivantes :

- Stockage — Lorsqu'un conteneur en cours d'exécution est arrêté et supprimé, le stockage des données dans ce conteneur disparaît, sauf si vous configurez un stockage plus permanent. Kubernetes prend en charge de nombreux types de stockage différents et les regroupe sous l'égide de Volumes. Les types de stockage incluent CephFS, NFS, iSCSI, etc. Vous pouvez même utiliser un périphérique de blocage local depuis l'ordinateur local. L'un de ces types de stockage étant disponible sur votre cluster, vous pouvez monter le volume de stockage sur un point de

montage sélectionné dans le système de fichiers de votre conteneur. Un volume persistant est un volume qui continue d'exister après la suppression du pod, tandis qu'un volume éphémère est supprimé lorsque le pod est supprimé. Si votre administrateur de cluster a créé différentes classes de stockage pour votre cluster, vous pouvez choisir les attributs du stockage que vous utilisez, par exemple si le volume est supprimé ou récupéré après utilisation, s'il sera étendu si davantage d'espace est nécessaire, et même s'il répond à certaines exigences de performance.

- Secrets — En mettant les secrets à la disposition des conteneurs dans les spécifications du Pod, vous pouvez fournir les autorisations dont ces conteneurs ont besoin pour accéder aux systèmes de fichiers, aux bases de données ou à d'autres actifs protégés. Les clés, les mots de passe et les jetons font partie des éléments qui peuvent être stockés en tant que secrets. L'utilisation de secrets vous évite d'avoir à stocker ces informations dans des images de conteneurs, mais vous n'avez qu'à les mettre à la disposition des conteneurs en cours d'exécution. Similaire à Secrets are ConfigMaps. A `ConfigMap` a tendance à contenir des informations moins critiques, telles que les paires clé-valeur pour configurer un service.
- Ressources de conteneur — Les objets permettant de configurer davantage les conteneurs peuvent prendre la forme d'une configuration de ressources. Pour chaque conteneur, vous pouvez demander la quantité de mémoire et de processeur qu'il peut utiliser, ainsi que fixer des limites à la quantité totale de ces ressources que le conteneur peut utiliser. Voir Gestion des ressources pour les pods et les conteneurs pour des exemples.
- Interruptions — Les pods peuvent être perturbés involontairement (un nœud tombe en panne) ou volontairement (une mise à niveau est souhaitée). En configurant un budget d'interruption du Pod, vous pouvez exercer un certain contrôle sur la disponibilité de votre application en cas de perturbations. Consultez la section Spécification d'un budget d'interruption pour votre application pour des exemples.
- Espaces de noms — Kubernetes propose différentes méthodes pour isoler les composants et les charges de travail Kubernetes les uns des autres. L'exécution de tous les pods d'une application particulière dans le même espace de noms est une méthode courante pour sécuriser et gérer ces pods ensemble. Vous pouvez créer vos propres espaces de noms à utiliser ou choisir de ne pas indiquer d'espace de noms (ce qui oblige Kubernetes à utiliser l'espace de noms). `default` Les composants du plan de contrôle Kubernetes s'exécutent généralement dans l'espace de noms du système Kube.

La configuration qui vient d'être décrite est généralement regroupée dans un fichier YAML à appliquer au cluster Kubernetes. Pour les clusters Kubernetes personnels, vous pouvez simplement stocker ces fichiers YAML sur votre système local. Cependant, avec des clusters et des charges de travail

plus critiques, GitOpsc'est un moyen populaire d'automatiser le stockage et les mises à jour des charges de travail et des ressources de l'infrastructure Kubernetes.

Les objets utilisés pour rassembler et déployer les informations du Pod sont définis par l'une des méthodes de déploiement suivantes.

Déploiement de pods

La méthode que vous choisirez pour déployer des pods dépend du type d'application que vous prévoyez d'exécuter avec ces pods. Voici certains de vos choix :

- Applications sans état : une application sans état n'enregistre pas les données de session d'un client. Une autre session n'a donc pas besoin de se référer à ce qui s'est passé lors d'une session précédente. Cela permet de remplacer plus facilement les pods par de nouveaux s'ils deviennent insalubres ou de les déplacer sans enregistrer leur état. Si vous exécutez une application sans état (telle qu'un serveur Web), vous pouvez utiliser un déploiement pour déployer des pods et ReplicaSets. A ReplicaSet définit le nombre d'instances d'un Pod que vous souhaitez exécuter simultanément. Bien que vous puissiez exécuter un pod ReplicaSet directement, il est courant d'exécuter des répliques directement dans un déploiement, afin de définir le nombre de répliques d'un pod à exécuter à la fois.
- Applications dynamiques — Une application dynamique est une application dans laquelle l'identité du Pod et l'ordre dans lequel les Pods sont lancés sont importants. Ces applications ont besoin d'un stockage persistant stable qui doit être déployé et mis à l'échelle de manière cohérente. Pour déployer une application dynamique dans Kubernetes, vous pouvez utiliser. StatefulSets Une base de données est un exemple d'application généralement exécutée en tant StatefulSet que. Dans un StatefulSet, vous pouvez définir les répliques, le Pod et ses conteneurs, les volumes de stockage à monter et les emplacements dans le conteneur où les données sont stockées. Voir Exécuter une application dynamique répliquée pour un exemple de déploiement d'une base de données en tant que. ReplicaSet
- Applications par nœud : il arrive que vous souhaitiez exécuter une application sur chaque nœud de votre cluster Kubernetes. Par exemple, votre centre de données peut exiger que chaque ordinateur exécute une application de surveillance ou un service d'accès à distance spécifique. Pour Kubernetes, vous pouvez utiliser a DaemonSetpour vous assurer que l'application sélectionnée s'exécute sur tous les nœuds de votre cluster.
- Les applications s'exécutent jusqu'à la fin : vous souhaitez exécuter certaines applications pour effectuer une tâche particulière. Cela peut inclure un système qui publie des rapports d'état mensuels ou nettoie les anciennes données. Un objet Job peut être utilisé pour configurer une application afin qu'elle démarre et s'exécute, puis qu'elle se ferme une fois la tâche terminée. Un

CronJobobjet vous permet de configurer une application pour qu'elle s'exécute à une heure, une minute, un jour du mois ou un jour de la semaine spécifiques, en utilisant une structure définie par le format crontab Linux.

Rendre les applications accessibles depuis le réseau

Les applications étant souvent déployées sous la forme d'un ensemble de microservices déplacés vers différents emplacements, Kubernetes avait besoin d'un moyen permettant à ces microservices de se retrouver entre eux. De plus, pour que d'autres puissent accéder à une application en dehors du cluster Kubernetes, Kubernetes avait besoin d'un moyen d'exposer cette application sur des adresses et des ports externes. Ces fonctionnalités liées au réseau sont réalisées avec les objets Service et Ingress, respectivement :

- Services — Étant donné qu'un Pod peut se déplacer vers différents nœuds et adresses, un autre Pod devant communiquer avec le premier Pod peut avoir du mal à le localiser. Pour résoudre ce problème, Kubernetes vous permet de représenter une application en tant que service. Avec un service, vous pouvez identifier un pod ou un ensemble de pods avec un nom particulier, puis indiquer quel port expose le service de cette application depuis le pod et quels ports une autre application pourrait utiliser pour contacter ce service. Un autre pod au sein d'un cluster peut simplement demander un service par son nom et Kubernetes dirigera cette demande vers le port approprié pour une instance du pod exécutant ce service.
- Entrée — L'entrée permet de mettre les applications représentées par Kubernetes Services à la disposition des clients extérieurs au cluster. Les fonctionnalités de base d'Ingress incluent un équilibreur de charge (géré par Ingress), le contrôleur Ingress et des règles de routage des demandes du contrôleur vers le Service. Il existe plusieurs contrôleurs d'entrée parmi lesquels vous pouvez choisir avec Kubernetes.

Étapes suivantes

Comprendre les concepts de base de Kubernetes et leur lien avec Amazon EKS vous aidera à parcourir à la fois la documentation Amazon EKS et la documentation Kubernetes pour trouver les informations dont vous avez besoin pour gérer les clusters Amazon EKS et déployer des charges de travail sur ces clusters. Pour commencer à utiliser Amazon EKS, choisissez l'une des options suivantes :

- the section called “Créer un cluster (eksctl)”
- the section called “Créer un cluster ”

- the section called "Exemple de déploiement (Linux)"
- Gestion du cluster

Déployez des clusters Amazon EKS dans des environnements cloud et sur site

Comprendre les options de déploiement d'Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) est un service Kubernetes entièrement géré qui vous permet d'exécuter Kubernetes de manière fluide dans le cloud et dans vos environnements sur site.

Dans le cloud, Amazon EKS automatise la gestion de l'infrastructure du cluster Kubernetes pour le plan de contrôle et les nœuds Kubernetes. Cela est essentiel pour planifier les conteneurs, gérer la disponibilité des applications, dimensionner les ressources de manière dynamique, optimiser le calcul, stocker les données des clusters et exécuter d'autres fonctions critiques. Avec Amazon EKS, vous bénéficiez des performances, de l'évolutivité, de la fiabilité et de la disponibilité robustes de l' AWS infrastructure, ainsi que d'intégrations natives avec les services de AWS réseau, de sécurité, de stockage et d'observabilité.

Pour simplifier l'exécution de Kubernetes dans vos environnements sur site, vous pouvez utiliser les mêmes clusters, fonctionnalités et outils Amazon EKS ou utiliser les mêmes nœuds hybrides the section called "Nœuds" Amazon EKS sur votre propre infrastructure, ou vous pouvez utiliser Amazon EKS Anywhere pour les environnements autonomes à airgap.

Amazon EKS dans le cloud

Vous pouvez utiliser Amazon EKS pour effectuer des calculs dans AWS les régions, les zones AWS locales et les zones de AWS longueur d'onde. Avec Amazon EKS dans le cloud, la sécurité, l'évolutivité et la disponibilité du plan de contrôle Kubernetes sont entièrement gérées par AWS la région. AWS Lorsque vous exécutez des applications de calcul dans des AWS régions, vous bénéficiez de l'ensemble des fonctionnalités d' AWS Amazon EKS, notamment le mode automatique d'Amazon EKS, qui automatise entièrement la gestion de l'infrastructure de clusters Kubernetes pour le calcul, le stockage et la mise en réseau en un seul clic AWS . Lorsque vous exécutez des applications avec du calcul dans AWS des zones locales et des zones de AWS longueur d'onde, vous pouvez utiliser les nœuds autogérés Amazon EKS pour connecter les EC2 instances Amazon au calcul de votre cluster et utiliser les autres AWS services disponibles dans les zones AWS locales

et les zones de AWS longueur d'onde. Pour plus d'informations, consultez les fonctionnalités AWS des zones locales et les fonctionnalités des zones de AWS Wavelength.

	Amazon EKS dans AWS les régions	Amazon EKS dans les zones locales/de longueur d'onde
Gestion du plan de contrôle Kuberenetes	AWS-géré	AWS-géré
Emplacement du plan de contrôle Kubernetes	AWS Régions	AWS Régions
Plan de données Kubernetes	• Mode automatique d'Amazon EKS • Groupes de nœuds gérés par Amazon EKS • EC2 Nœuds autogérés Amazon • AWS Fargate	• Groupes de nœuds gérés par Amazon EKS (Zones Locales uniquement) • EC2 Nœuds autogérés Amazon
Emplacement du plan de données Kubernetes	AWS Régions	AWS Zones locales ou de longueur d'onde

Amazon EKS dans votre centre de données ou vos environnements périphériques

Si vous devez exécuter des applications dans vos propres centres de données ou environnements périphériques, vous pouvez utiliser Amazon EKS sur AWS Outposts Amazon EKS Hybrid Nodes. Vous pouvez utiliser des nœuds autogérés avec des EC2 instances Amazon sur AWS Outposts pour le calcul de votre cluster, ou vous pouvez utiliser des nœuds hybrides Amazon EKS avec votre propre infrastructure sur site ou de périphérie pour le calcul de votre cluster. AWS Outposts est une infrastructure AWS gérée que vous gérez dans vos centres de données ou installations de colocation, tandis qu'Amazon EKS Hybrid Nodes s'exécute sur vos machines physiques ou virtuelles que vous gérez dans vos environnements locaux ou périphériques. Amazon EKS on AWS Outposts et Amazon EKS Hybrid Nodes nécessitent une connexion fiable entre vos environnements sur site et une AWS région, et vous pouvez utiliser les mêmes clusters, fonctionnalités et outils Amazon EKS

que ceux que vous utilisez pour exécuter des applications dans le cloud. Lors de l'exécution sur AWS Outposts, vous pouvez également déployer l'intégralité du cluster Kubernetes sur Outposts avec les clusters locaux Amazon EKS sur AWS Outposts. AWS

	Nœuds hybrides Amazon EKS	Amazon EKS sur AWS Outposts
Gestion du plan de contrôle Kuberenetes	AWS-géré	AWS-géré
Emplacement du plan de contrôle Kubernetes	AWS Régions	AWS Régions ou AWS Outposts
Plan de données Kubernetes	Machines physiques ou virtuelles gérées par le client	EC2 Nœuds autogérés Amazon
Emplacement du plan de données Kubernetes	Centre de données client ou environnement périphérique	Centre de données client ou environnement périphérique

Amazon EKS Anywhere pour les environnements isolés

Amazon EKS Anywhere simplifie la gestion des clusters Kubernetes grâce à l'automatisation de tâches lourdes et indifférenciées telles que la configuration de l'infrastructure et les opérations de cycle de vie des clusters Kubernetes dans les environnements sur site et de périphérie. Contrairement à Amazon EKS, Amazon EKS Anywhere est un produit géré par le client et les clients sont responsables des opérations du cycle de vie des clusters et de la maintenance des clusters Amazon EKS Anywhere. Amazon EKS Anywhere repose sur l'API de cluster (CAPI) du sous-projet Kubernetes et prend en charge toute une gamme d'infrastructures, notamment VMware vSphere, bare metal, Nutanix, Apache et Snow. CloudStack AWS Amazon EKS Anywhere peut être exécuté dans des environnements isolés et propose des intégrations facultatives avec des AWS services régionaux pour l'observabilité et la gestion des identités. Pour bénéficier de l'assistance relative à Amazon EKS Anywhere et de l'accès aux modules complémentaires AWS Kubernetes vendus, vous pouvez acheter des abonnements Amazon EKS Anywhere Enterprise.

	Amazon EKS Anywhere
Gestion du plan de contrôle Kuberenetes	Géré par le client

	Amazon EKS Anywhere
Emplacement du plan de contrôle Kubernetes	Centre de données client ou environnement périphérique
Plan de données Kubernetes	Machines physiques ou virtuelles gérées par le client
Emplacement du plan de données Kubernetes	Centre de données client ou environnement périphérique

Outillage Amazon EKS

Vous pouvez utiliser le connecteur Amazon EKS pour enregistrer et connecter tout cluster Kubernetes conforme AWS et le visualiser dans la console Amazon EKS. Une fois qu'un cluster est connecté, vous pouvez voir le statut, la configuration et les charges de travail de ce cluster dans la console Amazon EKS. Vous pouvez utiliser cette fonctionnalité pour afficher les clusters connectés dans la console Amazon EKS, mais le connecteur Amazon EKS n'active pas les opérations de gestion ou de mutation pour vos clusters connectés via la console Amazon EKS.

Amazon EKS Distro est la AWS distribution des composants Kubernetes sous-jacents qui alimentent toutes les offres Amazon EKS. Il inclut les composants de base nécessaires au bon fonctionnement d'un cluster Kubernetes, tels que les composants du plan de contrôle Kubernetes (etcd, kube-apiserver, kube-scheduler) et les composants réseau (CoreDNS, kube-proxy, plug-ins CNI). kube-controller-manager Amazon EKS Distro peut être utilisé pour gérer automatiquement les clusters Kubernetes avec les outils de votre choix. Les déploiements d'Amazon EKS Distro ne sont pas couverts par les plans de AWS support.

Configuration pour utiliser Amazon EKS

Pour préparer la gestion par ligne de commande de vos clusters Amazon EKS, vous devez installer plusieurs outils. Utilisez ce qui suit pour configurer les informations d'identification, créer et modifier des clusters, et utiliser les clusters une fois qu'ils sont en cours d'exécution :

- Configuration de la AWS CLI : obtenez la AWS CLI pour configurer et gérer les services dont vous avez besoin pour travailler avec les clusters Amazon EKS. En particulier, vous avez besoin d'une AWS CLI pour configurer les informations d'identification, mais vous en avez également besoin pour d'autres AWS services.
- Configurer kubectl et eksctl : la `eksctl` CLI interagit avec les clusters Amazon AWS EKS pour créer, modifier et supprimer. Une fois le cluster créé, utilisez la `kubectl` commande open source pour gérer les objets Kubernetes au sein de vos clusters Amazon EKS.
- Configurez un environnement de développement (facultatif) : pensez à ajouter les outils suivants :
  - Outil de déploiement local — Si vous débutez avec Kubernetes, envisagez d'installer un outil de déploiement local tel que minikube ou kind. Ces outils vous permettent de disposer d'un cluster Amazon EKS sur votre machine locale pour tester les applications.
  - Gestionnaire de packages — helm est un gestionnaire de packages populaire pour Kubernetes qui simplifie l'installation et la gestion de packages complexes. Avec Helm, il est plus facile d'installer et de gérer des packages tels que le AWS Load Balancer Controller sur votre cluster Amazon EKS.

Étapes suivantes

- Configurer la AWS CLI
- Configurer kubectl et eksctl
- Démarrage rapide : déploiement d'une application Web et stockage des données

Configuration de la AWS CLI

La AWS CLI est un outil de ligne de commande permettant de travailler avec AWS des services, notamment Amazon EKS. Il est également utilisé pour authentifier les utilisateurs ou les rôles IAM afin d'accéder au cluster Amazon EKS et à d'autres AWS ressources depuis votre machine locale. Pour approvisionner des ressources AWS depuis la ligne de commande, vous devez obtenir un ID

de clé d' AWS accès et une clé secrète à utiliser dans la ligne de commande. Vous devez ensuite configurer ces informations d'identification dans la AWS CLI. Si vous n'avez pas encore installé la AWS CLI, consultez la section Installer ou mettre à jour la dernière version de la AWS CLI dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.

Pour créer une clé d'accès

1. Connectez-vous au AWS Management Console.
2. Pour les comptes mono-utilisateur ou multi-utilisateurs :
   - Compte utilisateur unique — : : En haut à droite, choisissez votre nom AWS d'utilisateur pour ouvrir le menu de navigation. Par exemple, choisissez **`webadmin`**.
   - Compte multi-utilisateurs — : : Choisissez IAM dans la liste des services. Dans le tableau de bord IAM, sélectionnez Utilisateurs, puis choisissez le nom de l'utilisateur.
3. Choisissez Informations d'identification de sécurité.
4. Sous Clés d'accès, choisissez Créer une clé d'accès.
5. Choisissez Interface de ligne de commande (CLI), puis Suivant.
6. Choisissez Create access key (Créer une clé d'accès).
7. Choisissez Télécharger le fichier .csv.

Pour configurer la AWS CLI

Après avoir installé la AWS CLI, procédez comme suit pour la configurer. Pour plus d'informations, voir Configurer la AWS CLI dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.

1. Dans une fenêtre de terminal, entrez la commande suivante :

   ```
   aws configure
   ```

   Vous pouvez éventuellement configurer un profil nommé, tel que `--profile cluster-admin`. Si vous configurez un profil nommé dans la AWS CLI, vous devez toujours transmettre cet indicateur dans les commandes suivantes.
2. Entrez vos AWS informations d'identification. Par exemple :

   ```
   Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   ```

```
Default region name [None]: region-code
Default output format [None]: json
```

Pour obtenir un jeton de sécurité

Si nécessaire, exécutez la commande suivante pour obtenir un nouveau jeton de sécurité pour la AWS CLI. Pour plus d'informations, consultez le manuel get-session-tokende référence des commandes de la AWS CLI.

Par défaut, le jeton est valide pendant 15 minutes. Pour modifier le délai d'expiration de session par défaut, transmettez l'indicateur `--duration-seconds`. Par exemple :

```
aws sts get-session-token --duration-seconds 3600
```

Cette commande renvoie les informations d'identification de sécurité temporaires pour une session AWS CLI. Le résultat de la réponse devrait être le suivant :

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

Pour vérifier l'identité de l'utilisateur

Si nécessaire, exécutez la commande suivante pour vérifier les AWS informations d'identification de votre identité d'utilisateur IAM (par exemple*`ClusterAdmin`*) pour la session du terminal.

```
aws sts get-caller-identity
```

Cette commande renvoie l'Amazon Resource Name (ARN) de l'entité IAM configurée pour la AWS CLI. Vous devriez voir l'exemple de réponse suivant :

```
{
```

```
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws: iam::01234567890:user/ClusterAdmin"
}
```

Étapes suivantes

- Configurer kubectl et eksctl
- Démarrage rapide : déploiement d'une application Web et stockage des données

Configurez **kubectl** et **eksctl**

Une fois la AWS CLI installée, vous devez installer deux autres outils pour créer et gérer vos clusters Kubernetes :

- `kubectl`: L'outil de ligne de `kubectl` commande est le principal outil que vous utiliserez pour gérer les ressources au sein de votre cluster Kubernetes. Cette page explique comment télécharger et configurer le `kubectl` binaire correspondant à la version de votre cluster Kubernetes. Voir Installer ou mettre à jour kubectl.
- `eksctl`: L'outil de ligne de `eksctl` commande est conçu pour créer des clusters EKS dans le AWS cloud ou sur site (avec EKS Anywhere), ainsi que pour modifier et supprimer ces clusters. Voir Installer eksctl.

Installation ou mise à jour **kubectl**

Cette rubrique vous aide à télécharger et installer ou mettre à jour le fichier binaire `kubectl` sur votre appareil. Le fichier binaire est identique aux versions communautaires en amont. Le binaire n'est pas propre à Amazon EKS ou AWS. Suivez les étapes ci-dessous pour obtenir la version spécifique dont vous avez besoin, bien que de nombreux constructeurs se contentent `brew install kubectl` de l'installer. `kubectl`

> **Note**
>
> Vous devez utiliser une version `kubectl` qui se situe à une différence de version mineure près du plan de contrôle de votre cluster Amazon EKS. Par exemple, un client `kubectl` `1.32` doit utiliser des clusters `1.31`, `1.32` et `1.33` Kubernetes.

Étape 1 : Vérifiez si elle **`kubectl`** est installée

Déterminez si vous avez déjà `kubectl` installé sur votre appareil.

```
kubectl version --client
```

Si `kubectl` est installé dans le chemin d'accès de votre appareil, l'exemple de sortie comprend des informations similaires à celles qui suivent. Si vous souhaitez mettre à jour la version que vous avez actuellement installée avec une version plus récente, passez à l'étape suivante en vous assurant d'installer la nouvelle version au même emplacement que votre version actuelle.

```
Client Version: v1.31.X-eks-1234567
```

Si vous ne recevez aucune sortie, cela signifie soit que vous ne l'avez pas `kubectl` installé, soit qu'il n'est pas installé à un emplacement situé sur le chemin de votre appareil.

Étape 2 : Installation ou mise à jour **`kubectl`**

Installez ou mettez à `kubectl` jour l'un des systèmes d'exploitation suivants :

- the section called "macOS"
- the section called "Linux (amd64)"
- the section called "Linux (arm64)"
- the section called "Windows"

> **Note**
>
> Si les téléchargements vers votre AWS région sont lents à partir des AWS régions utilisées dans cette section, pensez CloudFront à configurer la mise en avant du contenu. Pour plus d'informations, consultez Commencer avec une CloudFront distribution de base.

macOS

Suivez les étapes ci-dessous pour procéder à l'installation `kubectl` sur macOS. Les étapes sont les suivantes :

- Choisir et télécharger le fichier binaire correspondant à la version de Kubernetes que vous souhaitez.
- Vérifiez éventuellement la somme de contrôle du binaire.
- Ajout de l'exécution aux autorisations du binaire.
- Copier le binaire dans un dossier de votre PATH.
- Ajoutez éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Téléchargez le fichier binaire correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.29`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
darwin/amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/
darwin/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
darwin/amd64/kubectl
```

2. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

   a. Téléchargez le `SHA-256` checksum correspondant à la version Kubernetes de votre cluster.

   - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

   - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

   - Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
darwin/amd64/kubectl.sha256
```

b. Vérifiez le total de contrôle `SHA-256` du fichier binaire téléchargé.

```
openssl sha1 -sha256 kubectl
```

c. Assurez-vous que le total de contrôle généré dans la sortie correspond à celui du fichier `kubectl.sha256` téléchargé.

3. Appliquez les autorisations d'exécution au fichier binaire.

```
chmod +x ./kubectl
```

4. Copiez le fichier binaire vers un dossier de votre `PATH`. Si vous avez déjà installé une version de `kubectl`, nous vous recommandons de créer un fichier `$HOME/bin/kubectl` et de vérifier que le fichier `$HOME/bin` est en premier dans votre `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facultatif) Ajoutez le chemin `$HOME/bin` à votre fichier d'initialisation du shell de façon à ce qu'il soit configuré lorsque vous ouvrez un shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Suivez les étapes ci-dessous pour effectuer l'installation `kubectl` sous Linux (amd64). Les étapes sont les suivantes :

- Choisir et télécharger le fichier binaire correspondant à la version de Kubernetes que vous souhaitez.
- Vérifiez éventuellement la somme de contrôle du binaire.
- Ajout de l'exécution aux autorisations du binaire.
- Copier le binaire dans un dossier de votre PATH.
- Ajoutez éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Téléchargez le `kubectl` fichier binaire correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/linux/
amd64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/linux/
amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/
amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/
amd64/kubectl
```

2. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

   a. Téléchargez la `SHA-256` somme de contrôle pour la version Kubernetes de votre cluster depuis Amazon S3 à l'aide de la commande correspondant à la plate-forme matérielle de votre appareil.

   - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

   - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

   - Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

b. Vérifiez le total de contrôle `SHA-256` du fichier binaire que vous avez téléchargé à l'aide de l'une des commandes suivantes.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```

c. Pour le premier, vous devriez voir`kubectl: OK`, pour le second, vous pouvez vérifier que la somme de contrôle générée dans la sortie correspond à la somme de contrôle du fichier téléchargé`kubectl.sha256`.

3. Appliquez les autorisations d'exécution au fichier binaire.

```
chmod +x ./kubectl
```

4. Copiez le fichier binaire vers un dossier de votre `PATH`. Si vous avez déjà installé une version de `kubectl`, nous vous recommandons de créer un fichier `$HOME/bin/kubectl` et de vérifier que le fichier `$HOME/bin` est en premier dans votre `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facultatif) Ajoutez le chemin `$HOME/bin` à votre fichier d'initialisation du shell de façon à ce qu'il soit configuré lorsque vous ouvrez un shell.

> **Note**
>
> Cette étape suppose que vous utilisiez le shell Bash ; si vous utilisez un autre shell, modifiez la commande pour utiliser votre propre fichier d'initialisation shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

Suivez les étapes ci-dessous pour effectuer l'installation `kubectl` sous Linux (arm64). Les étapes sont les suivantes :

- Choisir et télécharger le fichier binaire correspondant à la version de Kubernetes que vous souhaitez.
- Vérifiez éventuellement la somme de contrôle du binaire.
- Ajout de l'exécution aux autorisations du binaire.
- Copier le binaire dans un dossier de votre PATH.
- Ajoutez éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Téléchargez le `kubectl` fichier binaire correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/linux/
arm64/kubectl
```

- Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/linux/
arm64/kubectl
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/linux/
arm64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/linux/
arm64/kubectl
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/linux/
arm64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/linux/
arm64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/
arm64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/
arm64/kubectl
```

2. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

a. Téléchargez la `SHA-256` somme de contrôle pour la version Kubernetes de votre cluster depuis Amazon S3 à l'aide de la commande correspondant à la plate-forme matérielle de votre appareil.

- Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

b. Vérifiez le total de contrôle `SHA-256` du fichier binaire que vous avez téléchargé à l'aide de l'une des commandes suivantes.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```

c. Pour le premier, vous devriez voir`kubectl: OK`, pour le second, vous pouvez vérifier que la somme de contrôle générée dans la sortie correspond à la somme de contrôle du fichier téléchargé`kubectl.sha256`.

3. Appliquez les autorisations d'exécution au fichier binaire.

```
chmod +x ./kubectl
```

4. Copiez le fichier binaire vers un dossier de votre `PATH`. Si vous avez déjà installé une version de `kubectl`, nous vous recommandons de créer un fichier `$HOME/bin/kubectl` et de vérifier que le fichier `$HOME/bin` est en premier dans votre `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facultatif) Ajoutez le chemin `$HOME/bin` à votre fichier d'initialisation du shell de façon à ce qu'il soit configuré lorsque vous ouvrez un shell.

> **Note**
>
> Cette étape suppose que vous utilisiez le shell Bash ; si vous utilisez un autre shell, modifiez la commande pour utiliser votre propre fichier d'initialisation shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

Suivez les étapes ci-dessous pour procéder à l'installation `kubectl` sous Windows. Les étapes sont les suivantes :

- Choisir et télécharger le fichier binaire correspondant à la version de Kubernetes que vous souhaitez.
- Vérifiez éventuellement la somme de contrôle du binaire.
- Copier le binaire dans un dossier de votre PATH.
- Ajoutez éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Ouvrez un PowerShell terminal.
2. Téléchargez le `kubectl` fichier binaire correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.33`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/
     windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.32`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/
     windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.31`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/bin/
     windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.30`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/bin/
     windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.29`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/bin/
windows/amd64/kubectl.exe
```

- Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
windows/amd64/kubectl.exe
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
windows/amd64/kubectl.exe
```

3. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

   a. Téléchargez le `SHA-256` checksum correspondant à la version Kubernetes de votre cluster pour Windows.

   - Kubernetes `1.33`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.3/2025-08-03/bin/
windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.32`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.7/2025-08-03/bin/
windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.31`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.11/2025-08-03/
bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.30`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-08-03/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.29`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-08-03/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-08-03/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/
bin/windows/amd64/kubectl.exe.sha256
```

b. Vérifiez le total de contrôle `SHA-256` du fichier binaire téléchargé.

```
Get-FileHash kubectl.exe
```

c. Assurez-vous que le total de contrôle généré dans la sortie correspond à celui du fichier `kubectl.sha256` téléchargé. La PowerShell sortie doit être une chaîne de caractères équivalente en majuscules.

4. Copiez le fichier binaire vers un dossier de votre `PATH`. Si vous disposez déjà d'un répertoire dans votre `PATH` que vous utilisez pour les utilitaires de ligne de commande, copiez le fichier binaire dans ce répertoire. Sinon, effectuez les étapes suivantes.

   a. Créez un répertoire pour vos fichiers binaires de ligne de commande, par exemple `C:\bin`.

   b. Copiez le fichier binaire `kubectl.exe` vers votre nouveau répertoire.

   c. Modifiez la variable d'environnement `PATH` de votre utilisateur ou système pour ajouter le nouveau répertoire à votre variable `PATH`.

d. Fermez votre PowerShell terminal et ouvrez-en un nouveau pour récupérer la nouvelle `PATH` variable.

5. Après avoir installé `kubectl`, vous pouvez vérifier sa version.

```
kubectl version --client
```

6. Lors de la première installation`kubectl`, il n'est pas encore configuré pour communiquer avec un serveur. Nous aborderons cette configuration au besoin dans d'autres procédures. Si vous devez mettre à jour la configuration pour communiquer avec un cluster en particulier, vous pouvez exécuter la commande suivante. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

7. Envisagez de configurer la saisie automatique, qui vous permet d'utiliser la touche de tabulation pour exécuter `kubectl` les sous-commandes après avoir saisi les premières lettres. Voir Kubectl autocomplete dans la documentation de Kubernetes pour plus de détails.

Installer **`eksctl`**

La `eksctl` CLI est utilisée pour travailler avec des clusters EKS. Il automatise de nombreuses tâches individuelles. Consultez la section Installation dans la `eksctl` documentation pour obtenir des instructions d'installation`eksctl`. Pour Linux, utilisez les instructions UNIX.

Lorsque vous utilisez `eksctl` le principal de sécurité IAM que vous utilisez, vous devez disposer des autorisations nécessaires pour utiliser les rôles Amazon EKS IAM, les rôles liés à un service AWS CloudFormation, un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et utilisation de rôles liés à un service dans le guide de l'utilisateur IAM. Vous devez effectuer toutes les étapes de ce guide avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :

```
aws sts get-caller-identity
```

Étapes suivantes

- Démarrage rapide : déploiement d'une application Web et stockage des données

Démarrage rapide : déploiement d'une application Web et stockage des données

Déployer une application de jeu et conserver ses données sur Amazon EKS

Ce didacticiel de démarrage rapide explique les étapes à suivre pour déployer l'exemple d'application de jeu 2048 et conserver ses données sur un cluster Amazon EKS Auto Mode à l'aide de eksctl. Le mode automatique Amazon EKS automatise les tâches de routine relatives au stockage par blocs en cluster, à la mise en réseau, à l'équilibrage de charge et au dimensionnement automatique du calcul.

Au fur et à mesure de notre progression, nous vous expliquerons le processus de configuration du cluster. Amazon EKS Auto Mode automatisera les tâches de création d'un nœud à l'aide d'une instance EC2 gérée, de création d'un équilibreur de charge d'application et de création d'un volume EBS.

Dans l'ensemble, vous déploierez un exemple de charge de travail avec les annotations personnalisées requises pour une intégration complète aux AWS services.

Dans ce tutoriel

À l'aide du modèle de `eksctl` cluster ci-dessous, vous allez créer un cluster avec le mode automatique EKS pour le provisionnement automatique des nœuds.

Configuration du VPC Lorsque vous utilisez le modèle de cluster eksctl ci-dessous, eksctl crée automatiquement un Virtual Private Cloud (IPv4 VPC) pour le cluster. Par défaut, eksctl configure un VPC qui répond à toutes les exigences du réseau, en plus de créer des points de terminaison publics et privés.

Le mode automatique de gestion des instances EKS ajoute ou supprime dynamiquement des nœuds dans votre cluster EKS en fonction des demandes de vos applications Kubernetes.

Persistance des données Utilisez la capacité de stockage par blocs du mode automatique d'EKS pour garantir la persistance des données d'application, même dans les scénarios impliquant le redémarrage ou la défaillance du pod.

Accès aux applications externes Utilisez la fonctionnalité d'équilibrage de charge du mode automatique d'EKS pour configurer dynamiquement un Application Load Balancer (ALB).

Prérequis

Avant de commencer, assurez-vous que les conditions préalables suivantes sont configurées pour utiliser Amazon EKS :

- Configuration de la AWS CLI et configuration des informations d'identification
- Installer eksctl
- Installer kubectl

Pour de plus amples informations, veuillez consulter Configuration.

Configuration du cluster

Dans cette section, vous allez créer un cluster à l'aide du mode automatique EKS pour le provisionnement dynamique des nœuds.

Créez un `cluster-config.yaml` fichier et collez-y le contenu suivant. Remplacez `region-code` par une région valide, telle que `us-east-1` :

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: <region-code>

autoModeConfig:
  enabled: true
```

Nous sommes maintenant prêts à créer le cluster.

Créez le cluster Amazon EKS :

```
eksctl create cluster -f cluster-config.yaml
```

> **Important**
>
> Si vous n'utilisez pas eksctl pour créer le cluster, vous devez baliser manuellement les sous-réseaux VPC.

Créez IngressClass

Créez un Kubernetes `IngressClass` pour le mode automatique EKS. IngressClass définit la manière dont le mode automatique EKS gère les ressources d'entrée. Cette étape configure la capacité d'équilibrage de charge du mode automatique EKS. Lorsque vous créez des ressources d'entrée pour vos applications, le mode automatique d'EKS les utilise pour provisionner et gérer automatiquement les équilibreurs de charge, en intégrant vos applications Kubernetes aux services d'équilibrage de charge. IngressClass AWS

Enregistrez le fichier yaml suivant sous `ingressclass.yaml` le nom :

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: eks.amazonaws.com/alb
```

Appliquez le IngressClass à votre cluster :

```
kubectl apply -f ingressclass.yaml
```

Déployez l'exemple d'application de jeu 2048

Dans cette section, nous vous expliquons les étapes à suivre pour déployer le célèbre « jeu 2048 » en tant qu'exemple d'application au sein du cluster. Le manifeste fourni inclut des annotations personnalisées pour l'Application Load Balancer (ALB). Ces annotations s'intègrent à l'EKS et indiquent à celui-ci de gérer le trafic HTTP entrant comme « orienté vers Internet » et de l'acheminer vers le service approprié dans l'`game-2048`espace de noms en utilisant le type de cible « ip ».

> **Note**
>
> L'`docker-2048`image de l'exemple est une image de `x86_64` conteneur qui ne s'exécutera pas sur d'autres architectures.

1. Créez un espace de noms Kubernetes appelé `game-2048` avec le drapeau. `--save-config`

   ```
   kubectl create namespace game-2048 --save-config
   ```

   Le résultat de la réponse devrait être le suivant :

   ```
   namespace/game-2048 created
   ```

2. Déployez l'application 2048 Game Sample.

   ```
   kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
   ```

   Ce manifeste définit un déploiement, un service et une entrée Kubernetes pour l'espace de `game-2048` noms, créant ainsi les ressources nécessaires pour déployer et exposer l'application au sein du `game-2048` cluster. Cela inclut la création d'un service nommé `service-2048` qui expose le déploiement sur le port`80`, et d'une ressource d'entrée nommée `ingress-2048` qui définit les règles de routage pour le trafic HTTP entrant et les annotations pour un Application Load Balancer (ALB) connecté à Internet. Le résultat de la réponse devrait être le suivant :

   ```
   namespace/game-2048 configured
   deployment.apps/deployment-2048 created
   service/service-2048 created
   ingress.networking.k8s.io/ingress-2048 created
   ```

3. Exécutez la commande suivante pour obtenir la ressource Ingress pour l'espace de `game-2048` noms.

   ```
   kubectl get ingress -n game-2048
   ```

   Le résultat de la réponse devrait être le suivant :

```
NAME           CLASS   HOSTS   ADDRESS
                     PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

Vous devrez attendre plusieurs minutes le temps que l'Application Load Balancer (ALB) soit provisionné avant de commencer les étapes suivantes.

4. Ouvrez un navigateur Web et entrez le code `ADDRESS` de l'étape précédente pour accéder à l'application Web. Par exemple :

```
k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
```

Vous devriez voir le jeu 2048 dans votre navigateur. Jouez !

2048
SCORE
0
BEST
48
Join the numbers and get to the 2048 tile!
New Game
2
2

Persister les données à l'aide du mode automatique d'Amazon EKS

Maintenant que le jeu 2048 est opérationnel sur votre cluster Amazon EKS, il est temps de vous assurer que vos données de jeu sont conservées en toute sécurité grâce à la fonctionnalité de stockage par blocs du mode automatique d'Amazon EKS.

1. Créez un fichier nommé `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Appliquez le `StorageClass` :

```
kubectl apply -f storage-class.yaml
```

3. Créez une réclamation de volume persistante (PVC) pour demander le stockage de vos données de jeu. Créez un fichier nommé `ebs-pvc.yaml` et ajoutez-y le contenu suivant :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
  namespace: game-2048
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: auto-ebs-sc
```

4. Appliquez le PVC sur votre cluster :

```
kubectl apply -f ebs-pvc.yaml
```

Le résultat de la réponse devrait être le suivant :

```
persistentvolumeclaim/game-data-pvc created
```

5. Vous devez maintenant mettre à jour le déploiement de votre jeu 2048 pour utiliser ce PVC pour stocker des données. Le déploiement suivant est configuré pour utiliser le PVC pour stocker les données de jeu. Créez un fichier nommé `ebs-deployment.yaml` et ajoutez-y le contenu suivant :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3  # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

6. Appliquez le déploiement mis à jour :

```
kubectl apply -f ebs-deployment.yaml
```

Le résultat de la réponse devrait être le suivant :

```
deployment.apps/deployment-2048 configured
```

Grâce à ces étapes, votre jeu 2048 sur le cluster est désormais configuré pour conserver les données grâce à la capacité de stockage par blocs d'Amazon EKS Auto Mode. Cela garantit la sécurité de votre progression et de vos données dans le jeu, même en cas de défaillance d'un pod ou d'un nœud.

Si vous avez aimé ce didacticiel, faites-le nous savoir en nous faisant part de vos commentaires afin que nous puissions vous proposer d'autres didacticiels de démarrage rapide spécifiques à des cas d'utilisation, comme celui-ci.

Supprimer votre cluster et vos nœuds

Une fois que vous avez terminé d'utiliser le cluster que vous avez créé pour ce didacticiel, vous devez le nettoyer en le supprimant à l'aide de la commande suivante. Si vous souhaitez utiliser davantage ce cluster avant de le nettoyer, reportez-vous à la section Étapes suivantes.

```
eksctl delete cluster -f ./cluster-config.yaml
```

EKS nettoie automatiquement tous les nœuds qu'il a provisionnés lorsque le cluster est supprimé.

Découvrez Amazon EKS par l'exemple

Présentation

Ce guide de l'utilisateur Amazon EKS contient des procédures générales pour créer votre premier cluster EKS à partir de la ligne de commande ou constitue AWS Management Consoleune référence solide pour tous les principaux composants Amazon EKS. Toutefois, en tant qu'administrateur ou développeur de clusters Amazon EKS, vous pouvez mieux comprendre Amazon EKS en suivant les parcours d'apprentissage proposés sur des sites autres que ceux décrits dans ce guide. Ces sites peuvent vous aider à :

- Configurez des types spécifiques de clusters. Les types de clusters spécifiques peuvent être basés sur les types de charge de travail ou les exigences de sécurité. Par exemple, vous souhaiterez peut-être régler un cluster pour exécuter des charges de travail par lots, d'apprentissage automatique ou de calcul intensif.
- Améliorez vos clusters. Vous pouvez ajouter des fonctionnalités avancées à votre cluster pour fournir des éléments tels que l'observabilité, le stockage flexible, le dimensionnement automatique ou la mise en réseau de clusters spécialisés.
- Automatisez les mises à jour À l'aide de fonctionnalités telles que GitOps, vous pouvez configurer pour provisionner automatiquement l'infrastructure du cluster et les charges de travail, en fonction des modifications apportées à ces composants dans vos référentiels Git.
- Utilisez des outils de configuration de cluster avancés. Tout en `eksctl` fournissant un moyen rapide de créer un cluster, il existe d'autres outils qui peuvent faciliter la configuration et la mise à niveau de clusters plus complexes. Il s'agit notamment d'outils tels que Terraform et. CloudFormation

Pour commencer votre parcours d'apprentissage Amazon EKS, je vous recommande de consulter certains des sites décrits sur cette page. Si vous rencontrez des problèmes en cours de route, il existe également des ressources pour vous aider à les surmonter. Par exemple, le centre de connaissances Re:Post vous permet de rechercher dans la base de données d'assistance les problèmes de support liés à Amazon EKS. Le guide des meilleures pratiques d'Amazon EKS propose également des conseils sur les meilleures méthodes pour configurer vos clusters destinés à la production.

Atelier Amazon EKS

Partant d'une compréhension de base de Kubernetes et des conteneurs, l'atelier Amazon EKS est une plateforme d'apprentissage permettant à un administrateur de cluster de découvrir les fonctionnalités importantes d'Amazon EKS. Voici comment vous pouvez participer à l'atelier Amazon EKS :

- Amazon EKS Basics : regardez la vidéo sur la page d'introduction pour découvrir comment Amazon EKS implémente les fonctionnalités Kubernetes dans le cloud. AWS Si vous avez besoin d'une compréhension encore plus élémentaire de Kubernetes, regardez la vidéo Qu'est-ce que Kubernetes ?
- Configuration d'Amazon EKS : si vous avez un AWS compte, la section Configuration vous aide à configurer un CloudShell environnement pour créer un cluster. Il propose un choix entre eksctl (une simple ligne de commande de création de cluster) et Terraform (une infrastructure-as-code approche plus poussée pour créer un cluster) pour créer votre cluster Amazon EKS.
- Amazon EKS Pour démarrer : essayez une boutique en ligne simple dans la section Exemple d'application. Vous pouvez l'utiliser dans les autres exercices. Dans cette section, vous découvrirez également comment empaqueter des images de conteneurs et comment les microservices sont gérés à l'aide des pods, des déploiements, des services et des espaces de noms Kubernetes. StatefulSets Utilisez ensuite Kustomize pour déployer les modifications apportées aux manifestes Kubernetes.
- Principes fondamentaux d'Amazon EKS : à l'aide de AWS fonctionnalités telles que le AWS Load Balancer Controller, l'atelier vous montre comment exposer vos applications au monde extérieur. En ce qui concerne le stockage, l'atelier explique comment utiliser Amazon EBS pour le stockage par blocs, Amazon EFS pour le stockage de systèmes de fichiers et Amazon for NetApp ONTAP FSx pour gérer les systèmes de fichiers ONTAP dans. AWS Pour la gestion des nœuds, l'atelier vous aide à configurer des groupes de nœuds gérés.
- Fonctionnalités avancées d'Amazon EKS : les fonctionnalités les plus avancées proposées dans le cadre de l'atelier Amazon EKS incluent des ateliers de configuration :
  - Autoscaling : Cela inclut le dimensionnement automatique des nœuds (avec Cluster Autoscaler ou Karpenter) et le dimensionnement automatique de la charge de travail (avec Horizontal Pod Autoscaler et Cluster Proportional Autoscaler).
  - Observabilité : découvrez la journalisation OpenSearch, les informations sur les conteneurs sur Amazon EKS et la visibilité des coûts avec Kubecost dans un ensemble de laboratoires d'observabilité.

- Sécurité : cet ensemble de laboratoires de sécurité vous permet d'explorer la gestion des secrets, Amazon GuardDuty, les normes de sécurité des Pod et la gestion des politiques Kyverno.
- Mise en réseau : découvrez les fonctionnalités réseau d'Amazon EKS dans des laboratoires de mise en réseau qui incluent Amazon VPC CNI (prise en charge des plug-ins réseau) et Amazon VPC Lattice (pour configurer des clusters sur des comptes VC et utilisateurs).
- Automatisation : Labs on Automation vous explique les GitOpsméthodes de gestion de vos clusters et de vos projets, comme AWS Controllers for Kubernetes et Crossplane pour gérer les plans de contrôle Amazon EKS.

Tutoriels pratiques de configuration de clusters Amazon EKS

Un ensemble de didacticiels de configuration de clusters Amazon EKS disponibles sur le site AWS communautaire peut vous aider à créer des clusters Amazon EKS spécifiques et à améliorer ces clusters de différentes manières. Les didacticiels sont divisés en trois types différents :

Création de clusters

Ces didacticiels vous aident à créer des clusters qui peuvent être utilisés à des fins spécifiques. Ces objectifs spéciaux incluent la possibilité d'exécuter :

- Applications évolutives à l'échelle mondiale basées sur IPv6
- Tâches par lots asynchrones
- Microservices à fort trafic
- Autoscaling avec Karpenter sur Fargate
- Charges de travail financières
- Groupes de nœuds gérés par Windows

Améliorer les clusters

Une fois que vous avez un cluster existant, vous pouvez l'étendre et l'améliorer de manière à lui permettre d'exécuter des charges de travail spécialisées et d'améliorer les clusters. Ces didacticiels proposent des méthodes pour :

- Fournir des solutions de stockage avec EFS CSI

- Fournir un stockage de base de données dynamique avec EBS CSI
- Exposez les applications sur des IPv4 clusters à l'aide du AWS Load Balancer Controller
- Exposez les applications sur des IPv6 clusters à l'aide du AWS Load Balancer Controller

Optimisation AWS des services

À l'aide de ces didacticiels, vous pouvez mieux intégrer vos clusters aux AWS services. Ces didacticiels incluent ceux qui vous aideront à :

- Gérez les enregistrements DNS pour les microservices avec ExternalDNS
- Surveillez les applications avec CloudWatch
- Gérez les tâches asynchrones avec le stockage SQS et EFS
- Utiliser les AWS secrets de Secrets Manager provenant des charges de travail
- Configurer des MTL avec Fargate, NGINX et ACM PCA

Exemples Amazon EKS

Le référentiel Amazon EKS Samples stocke les manifestes à utiliser avec Amazon EKS. Ces manifestes vous permettent d'essayer différents types d'applications dans Amazon EKS ou de créer des types spécifiques de clusters Amazon EKS. Les exemples incluent des manifestes destinés à :

- Création d'un cluster AWS Amazon EKS Fargate
- Création d'un cluster avec un rôle IAM existant
- Ajouter un groupe de nœuds gérés par Ubuntu à un cluster
- Backup et restauration du stockage du pod avec des instantanés de volume
- Restaurez des volumes EBS montés comme PVCs avec plusieurs comptes
- Activer le protocole proxy pour le contrôleur d'entrée NGINX avec les équilibreurs de charge classiques
- Configurer la journalisation sur Fargate pour AWS OpenSearch
- Exécutez le SDK Python avec un fournisseur d'identité fédéré Web
- Déployer un exemple d'application sur un contrôleur NFS CSI
- Utilisez des instantanés de volume pour StatefulSets

- Déployez des pods sur des nœuds situés dans différentes zones de disponibilité

N'oubliez pas que ces échantillons sont uniquement destinés à l'apprentissage et aux tests et ne sont pas destinés à être utilisés en production.

AWS Tutoriels

Le site AWS Tutoriels publie quelques didacticiels Amazon EKS, mais propose également un outil de recherche pour trouver d'autres didacticiels publiés sur AWS des sites (tels que le site AWS communautaire). Les didacticiels Amazon EKS publiés directement sur ce site incluent :

- Déployer une application Web de conteneur sur Amazon EKS
- Exécutez des clusters Kubernetes à moindre coût (instances Amazon EKS et Spot)
- Comment optimiser les coûts des tâches Jenkins sur Kubernetes

Atelier pour développeurs

Si vous êtes développeur de logiciels et que vous souhaitez créer ou refactoriser des applications pour les exécuter sur Amazon EKS, l'atelier Amazon EKS Developers est un bon point de départ. L'atelier vous aide non seulement à créer des applications conteneurisées, mais également à déployer ces conteneurs dans un registre de conteneurs (ECR), puis dans un cluster Amazon EKS.

Commencez par l'atelier Amazon EKS Python pour suivre le processus de refactorisation d'une application python, puis configurez votre environnement de développement pour préparer le déploiement de l'application. Parcourez les sections sur les conteneurs, Kubernetes et Amazon EKS pour vous préparer à exécuter vos applications conteneurisées dans ces environnements.

Atelier Terraform

Bien qu'`eksctl`il s'agisse d'un outil simple pour créer un cluster, pour les infrastructure-as-code types plus complexes de déploiements Amazon EKS, Terraform est un outil de création et de gestion de clusters Amazon EKS populaire. L'atelier Terraform Amazon EKS explique comment utiliser Terraform pour créer un VPC AWS , créer des clusters Amazon EKS et ajouter des améliorations facultatives à votre cluster. En particulier, il existe une section pour créer un cluster Amazon EKS privé

AWS Formation Amazon EKS

AWS propose une formation officielle pour en savoir plus sur Amazon EKS. Une formation de trois jours intitulée Running Containers on Amazon Elastic Kubernetes Service enseigne :

- Principes fondamentaux de Kubernetes et Amazon EKS
- Comment créer des clusters Amazon EKS
- Sécurisation d'Amazon EKS avec l' AWS autorisation IAM et Kubernetes RBAC
- GitOps outils d'automatisation
- Outils de surveillance
- Techniques pour améliorer les coûts, l'efficacité et la résilience

Mise en route avec Amazon EKS

Assurez-vous que votre configuration vous permet d'utiliser Amazon EKS avant de consulter les guides de démarrage. Pour de plus amples informations, veuillez consulter Configuration.

Il existe deux guides de démarrage pour créer un nouveau cluster Kubernetes avec les nœuds dans Amazon EKS :

- Commencez avec Amazon EKS — eksctl — Ce guide de démarrage vous aide à installer toutes les ressources nécessaires pour commencer à utiliser Amazon EKS en utilisant `eksctl` un simple utilitaire en ligne de commande permettant de créer et de gérer des clusters Kubernetes sur Amazon EKS. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications. Il s'agit de la solution la plus simple et la plus rapide pour commencer à utiliser Amazon EKS.
- Démarrez avec Amazon EKS — AWS Management Console et la AWS CLI — Ce guide de démarrage vous aide à créer toutes les ressources nécessaires pour démarrer avec Amazon EKS à l'aide de l' AWS interface de ligne de commande AWS Management Console et. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications. Dans ce guide, vous créez manuellement chaque ressource requise pour un cluster Amazon EKS. Les procédures vous donnent une visibilité sur la façon dont chaque ressource est créée et la manière dont elles interagissent les unes avec les autres.

Nous proposons également les références suivantes :

- Pour une collection de didacticiels pratiques, voir Configuration du cluster EKS dans la AWS communauté.
- Pour des exemples de code, consultez la section Exemples de code pour Amazon EKS à l'aide d'Amazon EKS AWS SDKs.

Commencez avec Amazon EKS — Mode automatique EKS

Comme pour les autres expériences de démarrage d'EKS, la création de votre premier cluster avec le mode automatique EKS délègue la gestion du cluster lui-même à AWS. Cependant, le mode automatique d'EKS étend l'automatisation d'EKS en déléguant la responsabilité de nombreux services essentiels nécessaires à la configuration de l'infrastructure de charge de travail (nœuds,

réseaux et divers services), facilitant ainsi la gestion des nœuds et l'évolutivité pour répondre aux demandes de charge de travail.

Choisissez l'une des méthodes suivantes pour créer un cluster avec le mode automatique EKS :

- the section called " AWS CLI": utilisez l'interface de ligne de `aws` commande pour créer un cluster.
- the section called "Console de gestion": utilisez le AWS Management Console pour créer un cluster.
- the section called "CLI exsctl": utilisez l'interface de ligne de `eksctl` commande pour créer un cluster.

Si vous comparez différentes approches pour créer votre premier cluster EKS, sachez que le mode automatique EKS a AWS assumé des responsabilités supplémentaires en matière de gestion de cluster, notamment la configuration de composants pour :

- Démarrez et dimensionnez les nœuds à mesure que la charge de travail augmente et diminue.
- Mettez régulièrement à niveau le cluster lui-même (plan de contrôle), les systèmes d'exploitation des nœuds et les services exécutés sur les nœuds.
- Choisissez les paramètres par défaut qui déterminent des éléments tels que la taille et la vitesse du stockage des nœuds et la configuration du réseau Pod.

Pour plus de détails sur ce que vous pouvez obtenir avec les clusters en mode automatique EKS, consultezMode automatique EKS.

Commencez avec Amazon EKS — **`eksctl`**

> **Note**
>
> Cette rubrique explique comment démarrer sans le mode automatique EKS.
> Le mode automatique d'EKS automatise les tâches de routine pour le calcul en cluster, le stockage et la mise en réseau. Découvrez comment démarrer avec le mode automatique d'Amazon EKS.

Ce guide vous permet de créer toutes les ressources nécessaires pour commencer à utiliser Amazon Elastic Kubernetes Service (Amazon EKS) avec `eksctl`, un utilitaire de ligne de commande

simple pour créer et gérer les clusters Kubernetes sur Amazon EKS. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications.

Les procédures de ce guide créent automatiquement plusieurs ressources que vous devez créer manuellement lorsque vous créez votre cluster à l'aide de l'outil AWS Management Console. Si vous préférez créer manuellement la plupart des ressources pour mieux comprendre comment elles interagissent les unes avec les autres, utilisez le AWS Management Console pour créer votre cluster et effectuer des calculs. Pour de plus amples informations, veuillez consulter the section called "Création d'un cluster (console et CLI)".

Prérequis

Avant de commencer ce didacticiel, vous devez installer et configurer les outils AWS CLI, kubectl et eksctl comme décrit dans Configuration pour utiliser Amazon EKS.

Étape 1 : créer votre cluster Amazon EKS et votre nœud

> **Important**
>
> Pour démarrer le plus simplement et le plus rapidement possible, cette rubrique inclut les étapes pour créer un cluster et des nœuds avec des paramètres par défaut. Avant de créer un cluster et des nœuds pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres et de déployer un cluster et des nœuds avec les paramètres qui répondent à vos besoins. Pour plus d'informations, consultez the section called "Créer un cluster " et Gérer le calcul. Certains paramètres ne peuvent être activés que lors de la création de votre cluster et de vos nœuds.

Vous pouvez créer un cluster à l'aide de l'un des types de nœuds suivants. Pour en savoir plus sur chaque type, consultez Gérer le calcul. Une fois votre cluster déployé, vous pouvez ajouter d'autres types de nœuds.

- Fargate — Linux — Sélectionnez ce type de nœud si vous souhaitez exécuter des applications Linux sur. the section called " AWS Fargate" Fargate est un moteur de calcul sans serveur qui vous permet de déployer des pods Kubernetes sans gérer les instances Amazon. EC2
- Nœuds gérés — Linux — Sélectionnez ce type de nœud si vous souhaitez exécuter des applications Amazon Linux sur des EC2 instances Amazon. Bien que cela ne soit pas inclus dans

ce guide, vous pouvez également ajouter les nœuds autogérés par Windows et Bottlerocket à votre cluster.

Créez votre cluster Amazon EKS à l'aide de la commande suivante. Vous pouvez remplacer *my-cluster* par votre propre valeur. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster. Remplacez *region-code* par n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général.

Example

Fargate - Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes - Linux

```
eksctl create cluster --name my-cluster --region region-code
```

La création du cluster prend plusieurs minutes. Au cours de la création, vous verrez plusieurs lignes de sortie. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl`a créé un fichier de `kubectl` configuration `~/.kube/config` ou ajouté la configuration du nouveau cluster dans un fichier de configuration existant `~/.kube/config` sur votre ordinateur.

Une fois la création du cluster terminée, consultez la AWS CloudFormation pile nommée `eksctl-`*`my-cluster`*`-cluster` dans la AWS CloudFormation console pour voir toutes les ressources créées.

Étape 2 : Afficher les ressources Kubernetes

1. Affichez vos nœuds de cluster.

```
kubectl get nodes -o wide
```

L'exemple qui suit illustre un résultat.

Example

Fargate - Linux

```
NAME                                                STATUS   ROLES    AGE
 VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
                CONTAINER-RUNTIME
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
 8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
 7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes - Linux

```
NAME                                        STATUS   ROLES    AGE    VERSION
        INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
   CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
  v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
  v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Pour plus d'informations sur les éléments que vous voyez en sortie, consultez the section called "Accès aux ressources du cluster".

2. Affichez les applications en cours d'exécution sur votre cluster.

```
kubectl get pods -A -o wide
```

L'exemple qui suit illustre un résultat.

Example

Fargate - Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
          NODE                                                 NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
 192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
 192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

Managed nodes - Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
          NODE                                       NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345             1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
kube-system   aws-node-67890             1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-abcde   1/1     Running   0          14m
 192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          14m
 192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-12345           1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-67890           1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
```

Pour plus d'informations sur les éléments que vous voyez en sortie, consultez the section called "Accès aux ressources du cluster".

Étape 3 : supprimer votre cluster et vos nœuds

Une fois que vous avez terminé avec le cluster et les nœuds que vous avez créés pour ce didacticiel, vous devez nettoyer en supprimant le cluster et les nœuds à l'aide de la commande suivante. Si vous

souhaitez exécuter d'autres opérations avec ce cluster avant le nettoyage, consultez the section called "Étapes suivantes".

```
eksctl delete cluster --name my-cluster --region region-code
```

Étapes suivantes

Les rubriques suivantes de la documentation vous aideront à étendre les fonctionnalités de votre cluster.

- Déployez un exemple d'application sur votre cluster.
- Le principal IAM qui a créé le cluster est le seul principal capable de passer des appels au serveur d'API Kubernetes avec ou avec le. `kubectl` AWS Management Console Si vous souhaitez que d'autres principaux IAM aient accès à votre cluster, vous devez les ajouter. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Autorisations requises".
- Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres des clusters et des nœuds. Certains paramètres (tels que l'activation de l'accès SSH aux EC2 nœuds Amazon) doivent être définis lors de la création du cluster.
- Pour renforcer la sécurité de votre cluster, configurez le plugin Amazon VPC Container Networking Interface pour utiliser les rôles IAM pour les comptes de service.

Démarrez avec Amazon EKS AWS Management Console et la AWS CLI

> **Note**
>
> Cette rubrique explique comment démarrer sans le mode automatique EKS. Il utilise des groupes de nœuds gérés pour déployer des nœuds.
> Le mode automatique d'EKS automatise les tâches de routine pour le calcul en cluster, le stockage et la mise en réseau. Découvrez comment démarrer avec le mode automatique d'Amazon EKS. Le mode automatique EKS est la méthode préférée pour déployer des nœuds.

Ce guide vous aide à créer toutes les ressources nécessaires pour démarrer avec Amazon Elastic Kubernetes Service (Amazon EKS) à l'aide AWS Management Console de la CLI et de la CLI. AWS Dans ce guide, vous créez manuellement chaque ressource. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications.

Les procédures de ce guide vous donnent une visibilité complète sur la façon dont chaque ressource est créée et dont les ressources interagissent les unes avec les autres. Si vous préférez que la plupart des ressources soient créées automatiquement, utilisez la `eksctl` CLI pour créer votre cluster et vos nœuds. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster (eksctl)".

Prérequis

Avant de démarrer ce didacticiel, vous devez installer et configurer les outils et les ressources suivants dont vous avez besoin pour créer et gérer un cluster Amazon EKS.

- AWS CLI : outil de ligne de commande permettant de travailler avec AWS des services, notamment Amazon EKS. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande. Après avoir installé la AWS CLI, nous vous recommandons de la configurer également. Pour plus d'informations, consultez la section Configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. Notez que la AWS CLI v2 est requise pour utiliser l'option update-kubeconfig présentée sur cette page.
- **`kubectl`**— Un outil de ligne de commande permettant de travailler avec des clusters Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".
- Autorisations IAM requises : le principal de sécurité IAM que vous utilisez doit être autorisé à utiliser les rôles IAM Amazon EKS, les rôles liés à un service, AWS CloudFormation un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et utilisation de rôles liés à un service dans le guide de l'utilisateur IAM. Vous devez effectuer toutes les étapes de ce guide avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :

  ```
  aws sts get-caller-identity
  ```

Nous vous recommandons de terminer les étapes de cette rubrique dans un shell Bash. Si vous n'utilisez pas de shell Bash, certaines commandes de script telles que les caractères de continuation

de ligne et la façon dont les variables sont définies et utilisées nécessitent un ajustement de votre shell. En outre, les règles de votre shell en matière de guillemets peuvent être différentes. Pour plus d'informations, consultez la section Utilisation de guillemets avec des chaînes dans la AWS CLI du Guide de l'utilisateur de l'interface de ligne de AWS commande.

Étape 1 : créer votre cluster Amazon EKS

> **Important**
>
> Pour démarrer le plus simplement et le plus rapidement possible, cette rubrique inclut les étapes pour créer un cluster avec des paramètres par défaut. Avant de créer un cluster pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres et de déployer un cluster avec les paramètres qui répondent à vos besoins. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster ". Certains paramètres ne peuvent être activés que lors de la création de votre cluster.

1. Créez un Amazon VPC avec des sous-réseaux publics et privés qui répondent aux exigences Amazon EKS. Remplacez *region-code* par n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général. Vous pouvez remplacer *my-eks-vpc-stack* par n'importe quel nom que vous choisissez.

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

> **Tip**
>
> Pour obtenir la liste de toutes les ressources créées par la commande précédente, ouvrez la AWS CloudFormation console à l'adresse https://console.aws.amazon.com/cloudformation/. Choisissez la pile *my-eks-vpc-stack*, puis choisissez l'onglet Ressources.

2. Créez un rôle IAM de cluster et attachez-y la politique gérée par Amazon EKS IAM requise. Les clusters Kubernetes gérés par Amazon EKS appellent d'autres AWS services en votre nom afin de gérer les ressources que vous utilisez avec le service.

   a. Copiez le contenu suivant dans un fichier nommé *eks-cluster-role-trust-policy.json*.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "eks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Créez le rôle.

   ```
   aws iam create-role \
     --role-name myAmazonEKSClusterRole \
     --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
   ```

   c. Attachez la politique IAM gérée par Amazon EKS au rôle.

   ```
   aws iam attach-role-policy \
     --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
     --role-name myAmazonEKSClusterRole
   ```

3. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.

   Assurez-vous que la AWS région affichée dans le coin supérieur droit de votre console est la AWS région dans laquelle vous souhaitez créer votre cluster. Si ce n'est pas le cas, choisissez le menu déroulant à côté du nom de la AWS région et choisissez la AWS région que vous souhaitez utiliser.

4. Choisissez Créer un cluster. Si cette option ne s'affiche pas, choisissez d'abord Clusters dans le volet de navigation de gauche.

5. Sur la page Configure cluster (Configurer le cluster), procédez de la façon suivante :

a. Sélectionnez Configuration personnalisée et désactivez Utiliser le mode automatique EKS. (Si vous préférez un cluster en mode automatique EKS, reportez-vous plutôt à the section called "Console de gestion".)

b. Saisissez un Name (Nom) pour votre cluster, tel que `my-cluster`. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.

c. Pour Rôle de service de cluster, choisissez `myAmazonEKSClusterRole`.

d. Laissez les autres paramètres à leurs valeurs par défaut et choisissez Next (Suivant).

6. Sur la page Specify networking (Spécifier les réseaux), procédez comme suit :

   a. Sélectionnez l'ID du VPC que vous avez créé à l'étape précédente dans la liste déroulante des VPC. C'est quelque chose comme `* | my-eks-vpc-stack-VPC`.

   b. Choisissez les sous-réseaux créés lors d'une étape précédente dans la liste déroulante Sous-réseaux. Les sous-réseaux seront quelque chose comme `* | my-eks-vpc-stack-*`.

   c. Choisissez le groupe de sécurité créé lors d'une étape précédente dans la liste déroulante Groupes de sécurité supplémentaires. C'est quelque chose comme `* | my-eks-vpc-stack-ControlPlaneSecurityGroup-*`.

   d. Laissez les autres paramètres à leurs valeurs par défaut et choisissez Next (Suivant).

7. Sur la page Configurer l'observabilité, cliquez sur Suivant.

8. Sur la page Sélectionner des modules complémentaires, choisissez Suivant.

   Pour plus d'informations sur les modules complémentaires, consultez the section called "Modules complémentaires Amazon EKS".

9. Sur la page Configurer les paramètres des modules complémentaires, choisissez Suivant.

10. Sur la page Review and create (Vérifier et créer), choisissez Create (Créer).

    À droite du nom du cluster, le statut du cluster est Création pendant plusieurs minutes jusqu'à ce que le processus de provisionnement du cluster soit terminé. Ne passez pas à l'étape suivante tant que le statut n'est pas Actif.

> **Note**
>
> Vous pouvez recevoir un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si

cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

Étape 2 : configurer votre ordinateur de façon à ce qu'il communique avec votre cluster

Dans cette section, vous créez un fichier `kubeconfig` pour votre cluster. Les paramètres de ce fichier activent l'option `kubectl` CLI pour communiquer avec votre cluster.

Avant de continuer, assurez-vous que la création de votre cluster s'est bien terminée à l'étape 1.

1. Créez ou mettez à jour un fichier `kubeconfig` pour votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle vous avez créé votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Par défaut, le `config` fichier est créé dans `~/.kube` ou la configuration du nouveau cluster est ajoutée à un `config` fichier existant dans`~/.kube`.

2. Testez votre configuration.

   ```
   kubectl get svc
   ```

   > **Note**
   >
   > Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

   L'exemple qui suit illustre un résultat.

```
NAME              TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes    ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Étape 3 : créer des nœuds

> **Important**
>
> Pour démarrer le plus simplement et le plus rapidement possible, cette rubrique décrit les étapes permettant de créer des nœuds avec principalement des paramètres par défaut. Avant de créer des nœuds pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres et de déployer des nœuds avec les paramètres qui répondent à vos besoins. Pour de plus amples informations, veuillez consulter Gérer le calcul. Certains paramètres ne peuvent être activés que lors de la création de vos nœuds.

Cette procédure configure votre cluster pour utiliser des groupes de nœuds gérés pour créer des nœuds, en spécifiant les sous-réseaux et le rôle IAM du nœud que vous avez créés lors des étapes précédentes. Il vous permet d'exécuter des applications Amazon Linux sur des EC2 instances Amazon.

Pour en savoir plus sur les différentes manières de configurer les nœuds dans EKS, consultezGérer le calcul. Une fois votre cluster déployé, vous pouvez ajouter d'autres types de nœuds. Bien que cela ne soit pas inclus dans ce guide, vous pouvez également ajouter les nœuds autogérés par Windows et Bottlerocket à votre cluster.

Pour créer votre groupe de nœuds gérés par EC2 Linux

1. Créez un rôle IAM de nœud et attachez-y la politique gérée par Amazon EKS IAM requise. Le `kubelet` démon du nœud Amazon EKS effectue des appels AWS APIs en votre nom. Les nœuds reçoivent l'autorisation pour ces appels d'API via un profil d'instance IAM et les politiques associées.
   a. Copiez le contenu suivant dans un fichier nommé `node-role-trust-policy.json`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez le rôle IAM du nœud.

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

c. Attachez les politiques IAM gérées requises au rôle.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

d. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.

e. Choisissez le nom du cluster que vous avez créé à l'étape 1 : Créez votre cluster Amazon EKS, tel que *my-cluster*.

f. Sur la ***my-cluster*** page, effectuez les opérations suivantes :

g. Choisissez l'onglet Compute (Calcul).

h. Choisissez Add Node Group (Ajouter un groupe de nœuds).

2. Sur la page Configure Node Group (Configurer le groupe de nœuds), procédez comme suit :

   a. Pour Name (Nom), saisissez un nom unique pour votre groupe de nœuds gérés, tel que *my-nodegroup*. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

b. Pour le nom du rôle Node IAM, choisissez *myAmazonEKSNodeRole* le rôle que vous avez créé à l'étape précédente. Nous recommandons que chaque groupe de nœuds utilise son propre rôle IAM unique.

c. Choisissez Suivant.

3. Sur la page Set compute and scaling configuration (Définir la configuration de calcul et de mise à l'échelle), acceptez les valeurs par défaut et choisissez Next (Suivant).
4. Sur la page Specify networking (Spécifier les réseaux), acceptez les valeurs par défaut et choisissez Next (Suivant).
5. Sur la page Review and create (Vérifier et créer), vérifiez la configuration de votre groupe de nœuds gérés et choisissez Create (Créer).
6. Après quelques minutes, le Status (Statut) dans la Node Group configuration (Configuration du groupe de nœuds) passera de Creating (En cours de création) à Active (Actif). Ne passez pas à l'étape suivante tant que le statut n'est pas Actif.

Étape 4 : afficher les ressources

Vous pouvez afficher vos nœuds et vos applications Kubernetes.

1. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, choisissez le nom du cluster que vous avez créé, par exemple*my-cluster*.
2. Sur la ***my-cluster***page, sélectionnez les options suivantes :

   a. Onglet Compute (Calcul) – La liste Nodes (Nœuds) qui ont été déployés pour le cluster s'affiche. Vous pouvez choisir le nom d'un nœud pour voir plus d'informations à son sujet.

   b. Onglet Ressources : vous pouvez voir toutes les ressources Kubernetes déployées par défaut sur un cluster Amazon EKS. Sélectionnez un type de ressources dans la console pour en savoir plus.

Étape 5 : Suppression des ressources

Une fois que vous avez terminé d'utiliser le cluster et les nœuds que vous avez créés pour ce didacticiel, vous devez supprimer les ressources que vous avez créées. Si vous souhaitez exécuter d'autres opérations avec ce cluster de supprimer les ressources, consultez the section called “Étapes suivantes”.

1. Supprimez tous les profils de groupes de nœuds que vous avez créés.

a. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.

b. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, choisissez *my-cluster*.

c. Choisissez l'onglet Calcul.

d. Si vous avez créé un groupe de *my-nodegroup* nœuds, choisissez-le, puis choisissez Supprimer. Entrez *my-nodegroup*, puis choisissez Delete (Supprimer).

e. Ne continuez pas tant que les profils de groupes de nœuds ne sont pas supprimés.

2. Supprimez le cluster.

a. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, choisissez *my-cluster*.

b. Choisissez Supprimer le cluster.

c. Saisissez *my-cluster* (supprimer), puis sélectionnez Delete (Supprimer). Ne continuez pas tant que le cluster n'est pas supprimé.

3. Supprimez la AWS CloudFormation pile VPC que vous avez créée.

a. Ouvrez la CloudFormation console à l'adresse https://console.aws.amazon.com/cloudformation/.

b. Choisissez la pile *my-eks-vpc-stack*, puis choisissez Supprimer.

c. Dans la boîte *my-eks-vpc-stack* de dialogue de confirmation de suppression, choisissez Supprimer la pile.

4. Supprimez le rôle IAM que vous avez créé.

a. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.

b. Dans le volet de navigation de gauche, choisissez Rôles.

c. Sélectionnez chaque rôle que vous avez créé dans la liste (***myAmazonEKSClusterRole***, ainsi que*myAmazonEKSNodeRole*). Choisissez Delete (Supprimer), saisissez le texte de confirmation demandé, puis choisissez Delete (Supprimer).

Étapes suivantes

Les rubriques suivantes de la documentation vous aideront à étendre les fonctionnalités de votre cluster.

- Le principal IAM qui a créé le cluster est le seul principal capable de passer des appels au serveur d'API Kubernetes avec ou avec le. `kubectl` AWS Management Console Si vous souhaitez que d'autres principaux IAM aient accès à votre cluster, vous devez les ajouter. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Autorisations requises".
- Déployez un exemple d'application sur votre cluster.
- Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres des clusters et des nœuds. Certains paramètres (tels que l'activation de l'accès SSH aux EC2 nœuds Amazon) doivent être définis lors de la création du cluster.
- Pour renforcer la sécurité de votre cluster, configurez le plugin Amazon VPC Container Networking Interface pour utiliser les rôles IAM pour les comptes de service.

Automatisez l'infrastructure de clusters avec le mode automatique EKS

Le mode automatique d'EKS étend la AWS gestion des clusters Kubernetes au-delà du cluster lui-même, afin de AWS permettre également de configurer et de gérer l'infrastructure qui permet le bon fonctionnement de vos charges de travail. Vous pouvez déléguer les décisions clés en matière d'infrastructure et tirer parti de l'expertise de AWS for day-to-day operations. L'infrastructure de cluster gérée par AWS inclut de nombreuses fonctionnalités Kubernetes en tant que composants principaux, par opposition aux modules complémentaires, tels que le dimensionnement automatique du calcul, la mise en réseau des pods et des services, l'équilibrage de charge des applications, le DNS du cluster, le stockage par blocs et le support du GPU.

Pour commencer, vous pouvez déployer un nouveau cluster EKS Auto Mode ou activer le mode EKS Auto sur un cluster existant. Vous pouvez déployer, mettre à niveau ou modifier vos clusters en mode automatique EKS à l'aide d'eksctl, de la AWS AWS Management Console CLI, d'EKS APIs ou de vos outils préférés infrastructure-as-code.

Avec le mode automatique d'EKS, vous pouvez continuer à utiliser vos outils compatibles Kubernetes préférés. Le mode automatique d'EKS s'intègre à AWS des services tels qu'Amazon EC2, Amazon EBS et ELB, en tirant parti des ressources AWS cloud conformes aux meilleures pratiques. Ces ressources sont automatiquement dimensionnées, optimisées en termes de coûts et régulièrement mises à jour afin de minimiser les coûts opérationnels et les frais généraux.

Fonctionnalités

Le mode automatique EKS fournit les fonctionnalités de haut niveau suivantes :

Simplifiez la gestion des clusters Kubernetes : le mode automatique d'EKS rationalise la gestion d'EKS en fournissant des clusters prêts pour la production avec une charge opérationnelle minimale. Avec le mode automatique d'EKS, vous pouvez exécuter des charges de travail dynamiques et exigeantes en toute confiance, sans avoir besoin d'une expertise approfondie d'EKS.

Disponibilité des applications : le mode automatique EKS ajoute ou supprime dynamiquement des nœuds dans votre cluster EKS en fonction des demandes de vos applications Kubernetes. Cela réduit le besoin de planification manuelle des capacités et garantit la disponibilité des applications.

Efficacité : le mode automatique EKS est conçu pour optimiser les coûts de calcul tout en respectant la flexibilité définie par vos exigences NodePool et celles de votre charge de travail. Il met également

fin aux instances inutilisées et consolide les charges de travail sur d'autres nœuds afin d'améliorer la rentabilité.

Sécurité : le mode automatique EKS utilise AMIs des éléments considérés comme immuables pour vos nœuds. Ils AMIs renforcent le verrouillage des logiciels, activent les contrôles d'accès SELinux obligatoires et fournissent des systèmes de fichiers racine en lecture seule. En outre, les nœuds lancés par le mode automatique EKS ont une durée de vie maximale de 21 jours (que vous pouvez réduire), après quoi ils sont automatiquement remplacés par de nouveaux nœuds. Cette approche améliore votre niveau de sécurité en faisant régulièrement circuler les nœuds, conformément aux meilleures pratiques déjà adoptées par de nombreux clients.

Mises à niveau automatisées : le mode automatique d'EKS maintient votre cluster Kubernetes, vos nœuds et les composants associés à jour avec les derniers correctifs, tout en respectant les budgets d'interruption des modules (PDBs) et les budgets d'interruption () que vous NodePool avez configurés. NDBs Jusqu'à la durée de vie maximale de 21 jours, une intervention peut être nécessaire si le blocage PDBs ou d'autres configurations empêchent les mises à jour.

Composants gérés : Le mode automatique d'EKS inclut Kubernetes et les fonctionnalités AWS cloud en tant que composants de base qui devraient autrement être gérés en tant que modules complémentaires. Cela inclut la prise en charge intégrée des attributions d'adresses IP des pods, des politiques réseau des pods, des services DNS locaux, des plug-ins GPU, des vérificateurs de santé et du stockage EBS CSI.

Personnalisable NodePools et NodeClasses : si votre charge de travail nécessite des modifications des configurations de stockage, de calcul ou de mise en réseau, vous pouvez créer des configurations personnalisées NodePools à NodeClasses l'aide du mode automatique EKS. Bien que vous ne deviez pas modifier NodePools les configurations par défaut NodeClasses, vous pouvez ajouter de nouvelles configurations personnalisées NodePools ou en NodeClasses complément des configurations par défaut pour répondre à vos besoins spécifiques.

Composants automatisés

Le mode automatique EKS rationalise le fonctionnement de vos clusters Amazon EKS en automatisant les principaux composants de l'infrastructure. L'activation du mode automatique EKS réduit encore les tâches de gestion de vos clusters EKS.

Voici une liste des composants du plan de données automatisés :

- Calcul : pour de nombreuses charges de travail, le mode automatique d'EKS vous permet d'oublier de nombreux aspects du calcul de vos clusters EKS. Il s'agit des licences suivantes :
  - Nœuds : les nœuds du mode automatique EKS sont conçus pour être traités comme des appareils. Le mode automatique EKS effectue les opérations suivantes :
    - Choisit une AMI appropriée configurée avec de nombreux services nécessaires pour exécuter vos charges de travail sans intervention.
    - Verrouille l'accès aux fichiers de l'AMI à l'aide du mode SELinux coercitif et d'un système de fichiers racine en lecture seule.
    - Empêche l'accès direct aux nœuds en interdisant l'accès SSH ou SSM.
    - Inclut le support du GPU, avec des pilotes de noyau et des plugins distincts pour NVIDIA et Neuron GPUs, permettant des charges de travail de haute performance.
    - Gère automatiquement les avis d'interruption et les événements liés à l'état de santé de l' EC2 instance EC2 Spot
  - Mise à l'échelle automatique : s'appuyant sur le dimensionnement automatique de Karpenter, le mode automatique d'EKS surveille les pods non planifiables et permet de déployer de nouveaux nœuds pour exécuter ces pods. Lorsque les charges de travail sont interrompues, le mode automatique d'EKS perturbe et arrête les nœuds de manière dynamique lorsqu'ils ne sont plus nécessaires, optimisant ainsi l'utilisation des ressources.
  - Mises à niveau : La prise de contrôle de vos nœuds rationalise la capacité d'EKS Auto Mode à fournir des correctifs de sécurité et à mettre à niveau le système d'exploitation et les composants selon les besoins. Ces mises à niveau sont conçues pour perturber le moins possible vos charges de travail. Le mode automatique EKS impose une durée de vie maximale des nœuds de 21 jours pour garantir que le up-to-date logiciel et. APIs
- Équilibrage de charge : EKS Auto Mode rationalise l'équilibrage de charge en s'intégrant au service Elastic Load Balancing d'Amazon, en automatisant le provisionnement et la configuration des équilibreurs de charge pour les services Kubernetes et les ressources d'entrée. Il prend en charge les fonctionnalités avancées des équilibreurs de charge applicatifs et réseau, gère leur cycle de vie et les adapte aux demandes des clusters. Cette intégration fournit une solution d'équilibrage de charge prête à être utilisée en production, conforme aux AWS meilleures pratiques, vous permettant de vous concentrer sur les applications plutôt que sur la gestion de l'infrastructure.
- Stockage : le mode automatique d'EKS configure le stockage éphémère pour vous en configurant les types de volumes, les tailles de volume, les politiques de chiffrement et les politiques de suppression lors de la fermeture du nœud.

- Mise en réseau : le mode automatique d'EKS automatise les tâches réseau critiques pour la connectivité des pods et des services. Cela inclut le IPv6 support IPv4/et l'utilisation de blocs CIDR secondaires pour étendre les espaces d'adresses IP.
- Identity and Access Management : il n'est pas nécessaire d'installer l'agent EKS Pod Identity sur les clusters EKS Auto Mode.

Pour plus d'informations sur ces composants, consultez the section called "Comment ça marche".

Configuration

Bien que le mode automatique d'EKS gère efficacement la plupart des services de votre plan de données sans intervention de votre part, il peut arriver que vous souhaitiez modifier le comportement de certains de ces services. Vous pouvez modifier la configuration de vos clusters en mode automatique EKS de la manière suivante :

- Kubernetes DaemonSets : plutôt que de modifier les services installés sur vos nœuds, vous pouvez utiliser des daemonsets Kubernetes. Les daemonsets sont conçus pour être gérés par Kubernetes, mais s'exécutent sur tous les nœuds du cluster. De cette façon, vous pouvez ajouter des services spéciaux pour surveiller ou surveiller vos nœuds.
- NodePools Personnalisés et NodeClasses : Par défaut NodePools et NodeClasses configurés par le mode automatique d'EKS, vous ne devez pas les modifier. Pour personnaliser le comportement des nœuds, vous pouvez créer des cas supplémentaires NodePools ou NodeClasses pour des cas d'utilisation tels que :
  - Sélection de types d'instances spécifiques (par exemple, processeurs accélérés ou instances EC2 Spot).
  - Isolation des charges de travail à des fins de sécurité ou de suivi des coûts.
  - Configuration des paramètres de stockage éphémères tels que les IOPS, la taille et le débit.
- Équilibrage de charge : Certains services, tels que l'équilibrage de charge, qu'EKS Auto Mode exécute en tant qu'objets Kubernetes, peuvent être configurés directement sur vos clusters EKS Auto Mode.

Pour plus d'informations sur les options de configuration du mode automatique EKS, consultezthe section called "Configuration".

Modèle de responsabilité partagée

Le modèle de responsabilité AWS partagée définit les responsabilités en matière de sécurité et de conformité entre AWS et les clients. Les images et le texte ci-dessous comparent et contrastent les différences entre le client et les AWS responsabilités entre le mode automatique d'EKS et le mode standard d'EKS.

Le mode automatique d'EKS transfère une grande partie de la responsabilité partagée de l'infrastructure Kubernetes des clients vers. AWS Avec le mode automatique d'EKS, AWS vous assumez davantage la responsabilité de la sécurité du cloud, qui relevait autrefois de la responsabilité du client et qui est désormais partagée. Les clients peuvent désormais se concentrer davantage sur leurs applications tout en AWS gérant l'infrastructure sous-jacente.

Responsabilité du client

Dans le cadre du mode automatique d'EKS, les clients restent responsables des conteneurs d'applications, y compris de la disponibilité, de la sécurité et de la surveillance. Ils contrôlent également l'infrastructure VPC et la configuration du cluster EKS. Ce modèle permet aux clients de se concentrer sur les préoccupations spécifiques aux applications tout en déléguant la gestion de l'infrastructure du cluster à. AWS Des fonctionnalités optionnelles par nœud peuvent être incluses dans les clusters par le biais de AWS modules complémentaires.

AWS responsabilité

Avec le mode automatique d'EKS, ses responsabilités AWS s'étendent pour inclure la gestion de plusieurs composants critiques supplémentaires par rapport à ceux déjà gérés dans les clusters EKS n'utilisant pas le mode automatique. En particulier, le mode automatique d'EKS prend en charge la configuration, la gestion, la sécurité et le dimensionnement des EC2 instances lancées, ainsi que les fonctionnalités du cluster pour l'équilibrage de charge, la gestion des adresses IP, la politique réseau et le stockage par blocs. Les composants suivants sont gérés par AWS le mode automatique EKS :

- EC2 Instances lancées en mode automatique : AWS gère le cycle de vie complet des nœuds en tirant parti des instances EC2 gérées par Amazon. EC2 les instances gérées sont responsables de la configuration, de l'application des correctifs, de la surveillance et de la maintenance du système d'exploitation. Dans ce modèle, l'instance elle-même et le système d'exploitation client qui s'exécute sur celle-ci relèvent de la responsabilité de AWS. Les nœuds utilisent des variantes de Bottlerocket AMIs optimisées pour exécuter des conteneurs. Les Bottlerocket AMIs disposent d'un logiciel verrouillé, de systèmes de fichiers racines immuables et d'un accès réseau sécurisé (pour empêcher les communications directes via SSH ou SSM).
- Capacités du cluster : AWS gère le dimensionnement automatique du calcul, la mise en réseau des pods avec l'application des politiques réseau, l'intégration d'Elastic Load Balancing et la configuration des pilotes de stockage.
- Plan de contrôle du cluster : AWS continue de gérer le serveur d'API Kubernetes, les comptes multiples et la base de données etcd ENIs, comme avec le système EKS standard.
- Services de base et infrastructure globale : assume AWS la responsabilité des services sous-jacents de calcul, de stockage, de mise en réseau et de surveillance, ainsi que de l'infrastructure globale des régions, des zones locales et des emplacements périphériques.

Création d'un cluster avec le mode automatique d'Amazon EKS

Ce chapitre explique comment créer un cluster Amazon EKS avec le mode automatique activé à l'aide de divers outils et interfaces. Le mode automatique simplifie la création de clusters en configurant et en gérant automatiquement l'infrastructure de calcul, de réseau et de stockage du cluster. Vous allez apprendre à créer un cluster en mode automatique à l'aide de la AWS CLI ou de l'outil de ligne de commande eksctl. AWS Management Console

> **Note**
>
> Le mode automatique d'EKS nécessite la version 1.29 ou supérieure de Kubernetes. Le mode automatique EKS n'est pas disponible dans les `mx-central-1` AWS régions `ap-southeast-7` OR.

Choisissez votre outil préféré en fonction de vos besoins : il AWS Management Console fournit une interface visuelle idéale pour découvrir les fonctionnalités du mode automatique d'EKS et créer des clusters individuels. La AWS CLI convient parfaitement aux tâches de script et d'automatisation, en particulier lors de l'intégration de la création de clusters dans des flux de travail ou des pipelines

CI/CD existants. La CLI eksctl offre une expérience native à Kubernetes et est recommandée aux utilisateurs familiarisés avec les outils Kubernetes qui souhaitent des opérations de ligne de commande simplifiées avec des valeurs par défaut raisonnables.

Avant de commencer, assurez-vous que les prérequis nécessaires sont installés et configurés, y compris les autorisations IAM appropriées pour créer des clusters EKS. Pour savoir comment installer des outils CLI tels que `kubectlaws`, et`eksctl`, voirConfiguration.

Vous pouvez utiliser la AWS CLI AWS Management Console, ou la CLI eksctl pour créer un cluster avec le mode automatique Amazon EKS.

Rubriques

- Créez un cluster en mode automatique EKS avec la CLI eksctl
- Création d'un cluster en mode automatique EKS à l'aide de la AWS CLI
- Créez un cluster en mode automatique EKS avec le AWS Management Console

Créez un cluster en mode automatique EKS avec la CLI eksctl

Cette rubrique explique comment créer un cluster en mode automatique Amazon EKS à l'aide de l'interface de ligne de commande (CLI) eksctl. Vous pouvez créer un cluster en mode automatique en exécutant une seule commande CLI ou en appliquant un fichier de configuration YAML. Les deux méthodes fournissent les mêmes fonctionnalités, l'approche YAML offrant un contrôle plus précis des paramètres du cluster.

La CLI eksctl simplifie le processus de création et de gestion des clusters en mode automatique EKS en gérant la création et la configuration AWS des ressources sous-jacentes. Avant de continuer, assurez-vous que les AWS informations d'identification et les autorisations nécessaires sont configurées sur votre ordinateur local. Ce guide part du principe que vous connaissez les concepts de base d'Amazon EKS et que vous avez déjà installé les outils CLI requis.

> **Note**
>
> Vous devez installer la version `0.195.0` ou supérieure d'eksctl. Pour plus d'informations, consultez les versions d'eksctl sur. GitHub

Création d'un cluster en mode automatique EKS à l'aide d'une commande CLI

Vous devez avoir installé `aws` les `eksctl` outils et. Vous devez être connecté à la AWS CLI avec des autorisations suffisantes pour gérer les AWS ressources, notamment les EC2 instances, le EC2 réseau, les clusters EKS et les rôles IAM. Pour de plus amples informations, veuillez consulter Configuration.

Exécutez la commande suivante pour créer un nouveau cluster EKS Auto Mode avec

```
eksctl create cluster --name=<cluster-name> --enable-auto-mode
```

Création d'un cluster en mode automatique EKS avec un fichier YAML

Vous devez avoir installé `aws` les `eksctl` outils et. Vous devez être connecté à la AWS CLI avec des autorisations suffisantes pour gérer les AWS ressources, notamment les EC2 instances, le EC2 réseau, les clusters EKS et les rôles IAM. Pour de plus amples informations, veuillez consulter Configuration.

Passez en revue les options de configuration du mode automatique EKS dans l'exemple de ClusterConfig ressource ci-dessous. Pour la ClusterConfig spécification complète, consultez la documentation eksctl.

AWS suggère d'activer le mode automatique EKS. Si c'est la première fois que vous créez un cluster en mode automatique EKS, laissez la valeur `nodeRoleARN` non spécifiée pour créer un rôle Node IAM pour le mode automatique EKS. Si vous avez déjà un rôle Node IAM dans votre AWS compte, AWS suggère de le réutiliser.

AWS suggère de ne pas spécifier de valeur pour`nodePools`. Le mode automatique EKS créera des pools de nœuds par défaut. Vous pouvez utiliser l'API Kubernetes pour créer des pools de nœuds supplémentaires.

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: <cluster-name>
  region: <aws-region>
```

```
iam:
  # ARN of the Cluster IAM Role
  # optional, eksctl creates a new role if not supplied
  # suggested to use one Cluster IAM Role per account
  serviceRoleARN: <arn-cluster-iam-role>

autoModeConfig:
  # defaults to false
  enabled: boolean
  # optional, defaults to [general-purpose, system].
  # suggested to leave unspecified
  # To disable creation of nodePools, set it to the empty array ([]).
  nodePools: []string
  # optional, eksctl creates a new role if this is not supplied
  # and nodePools are present.
  nodeRoleARN: string
```

Enregistrez le `ClusterConfig` fichier sous `cluster.yaml` et utilisez la commande suivante pour créer le cluster :

```
eksctl create cluster -f cluster.yaml
```

Création d'un cluster en mode automatique EKS à l'aide de la AWS CLI

Les clusters en mode automatique EKS automatisent les tâches de gestion de routine des clusters pour le calcul, le stockage et la mise en réseau. Par exemple, les clusters en mode automatique EKS détectent automatiquement lorsque des nœuds supplémentaires sont nécessaires et fournissent de nouvelles EC2 instances pour répondre aux demandes de charge de travail.

Cette rubrique explique comment créer un nouveau cluster en mode automatique EKS à l'aide de la AWS CLI et éventuellement déployer un exemple de charge de travail.

Prérequis

- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
  - Connectez-vous à la CLI avec des autorisations IAM suffisantes pour créer des AWS ressources, notamment des politiques IAM, des rôles IAM et des clusters EKS.

- L'outil de ligne de commande kubectl installé sur votre appareil. AWS suggère que vous utilisiez la même version de kubectl que la version Kubernetes de votre cluster EKS. Pour installer ou mettre à jour kubectl, consultez. the section called "Configurer kubectl et eksctl"

Spécifier les sous-réseaux VPC

Amazon EKS Auto Mode déploie des nœuds sur des sous-réseaux VPC. Lorsque vous créez un cluster EKS, vous devez spécifier les sous-réseaux VPC dans lesquels les nœuds seront déployés. Vous pouvez utiliser les sous-réseaux VPC par défaut de votre AWS compte ou créer un VPC dédié aux charges de travail critiques.

- AWS suggère de créer un VPC dédié pour votre cluster. Découvrez comment the section called "Création d'un VPC".
- La console EKS aide à créer un nouveau VPC. Découvrez comment the section called "Console de gestion".
- Vous pouvez également utiliser le VPC par défaut de votre AWS compte. Suivez les instructions ci-dessous pour trouver le sous-réseau IDs.

Pour trouver le sous-réseau IDs de votre VPC par défaut

À l'aide de la AWS CLI :

1. Exécutez la commande suivante pour répertorier le VPC par défaut et ses sous-réseaux :

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs
 --query 'Vpcs[?IsDefault==`true`].VpcId' --output text)" --query 'Subnets[*].
{ID:SubnetId,AZ:AvailabilityZone}' --output table
```

2. Enregistrez la sortie et notez le sous-réseau IDs.

   Exemple de sortie :

```
----------------------------------------
|             DescribeSubnets          |
----------------------------------------
|   SubnetId        |   AvailabilityZone  |
|--------------------|---------------------|
|   subnet-012345678 |   us-west-2a        |
|   subnet-234567890 |   us-west-2b        |
```

```
|   subnet-345678901 |   us-west-2c        |
----------------------------------------
```

Rôles IAM pour les clusters en mode automatique EKS

Rôle IAM du cluster

Le mode automatique d'EKS nécessite un rôle IAM de cluster pour effectuer des actions sur votre AWS compte, telles que le provisionnement de nouvelles EC2 instances. Vous devez créer ce rôle pour accorder à EKS les autorisations nécessaires. AWS recommande d'associer les politiques AWS gérées suivantes au rôle IAM du cluster :

- EKSComputePolitique d'Amazon
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- EKSNetworkingPolitique d'Amazon
- EKSClusterPolitique d'Amazon

Rôle IAM du nœud

Lorsque vous créez un cluster en mode automatique EKS, vous spécifiez un rôle Node IAM. Lorsque le mode automatique d'EKS crée des nœuds pour traiter les charges de travail en attente, le rôle Node IAM est attribué à chaque nouveau nœud d' EC2 instance. Ce rôle permet au nœud de communiquer avec EKS mais n'est généralement pas accessible aux charges de travail exécutées sur le nœud.

Si vous souhaitez accorder des autorisations aux charges de travail exécutées sur un nœud, utilisez EKS Pod Identity. Pour de plus amples informations, veuillez consulter the section called “Identité du pod”.

Vous devez créer ce rôle et y associer la politique AWS gérée suivante :

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Rôle lié à un service

Le mode automatique EKS nécessite également un rôle lié au service, qui est automatiquement créé et configuré par. AWS Pour plus d'informations, consultez AWSServiceRoleForAmazonEKS.

Création d'un rôle IAM de cluster en mode automatique EKS

Étape 1 : Création de la politique de confiance

Créez une politique de confiance qui autorise le service Amazon EKS à assumer ce rôle. Enregistrez la politique sous la forme suivante `trust-policy.json` :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Étape 2 : Création du rôle IAM

Utilisez la politique de confiance pour créer le rôle IAM du cluster :

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Étape 3 : notez l'ARN du rôle

Récupérez et enregistrez l'ARN du nouveau rôle pour l'utiliser lors des étapes suivantes :

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Étape 4 : Joindre les politiques requises

Associez les politiques AWS gérées suivantes au rôle IAM du cluster pour accorder les autorisations nécessaires :

EKSClusterPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputePolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBlock StoragePolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

Amazon EKSLoad BalancingPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Création d'un rôle IAM pour un nœud en mode automatique EKS

Étape 1 : Création de la politique de confiance

Créez une politique de confiance qui autorise le service Amazon EKS à assumer ce rôle. Enregistrez la politique sous la forme suivante `node-trust-policy.json` :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Étape 2 : créer le rôle Node IAM

Utilisez le fichier node-trust-policy.json de l'étape précédente pour définir les entités qui peuvent assumer le rôle. Exécutez la commande suivante pour créer le rôle Node IAM :

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Étape 3 : notez l'ARN du rôle

Après avoir créé le rôle, récupérez et enregistrez l'ARN du rôle Node IAM. Vous aurez besoin de cet ARN dans les étapes suivantes. Utilisez la commande suivante pour obtenir l'ARN :

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Étape 4 : Joindre les politiques requises

Associez les politiques AWS gérées suivantes au rôle Node IAM pour fournir les autorisations nécessaires :

Amazon EKSWorker NodeMinimalPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Amazon EC2 ContainerRegistryPullOnly :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Création d'un cluster en mode automatique EKS

Présentation

Pour créer un cluster en mode automatique EKS à l'aide de la AWS CLI, vous aurez besoin des paramètres suivants :

- `cluster-name`: nom du cluster.
- `k8s-version`: La version de Kubernetes (par exemple, 1.31).
- `subnet-ids`: sous-réseau IDs identifié lors des étapes précédentes.
- `cluster-role-arn`: ARN du rôle IAM du cluster.
- `node-role-arn`: ARN du rôle IAM du nœud.

Configurations de cluster par défaut

Passez en revue ces valeurs et fonctionnalités par défaut avant de créer le cluster :

- `nodePools`: Le mode automatique EKS inclut les pools de nœuds à usage général et les pools de nœuds par défaut du système. En savoir plus sur les pools de nœuds.

Remarque : les pools de nœuds en mode automatique EKS sont différents des groupes de nœuds gérés par Amazon EKS mais peuvent coexister dans le même cluster.

- `computeConfig.enabled`: automatise les tâches de calcul de routine, telles que la création et la suppression d' EC2 instances.
- `kubernetesNetworkConfig.elasticLoadBalancing.enabled`: automatise les tâches d'équilibrage de charge, notamment la création et la suppression d'Elastic Load Balancers.
- `storageConfig.blockStorage.enabled`: automatise les tâches de stockage, telles que la création et la suppression de volumes Amazon EBS.
- `accessConfig.authenticationMode`: Nécessite des entrées d'accès EKS. En savoir plus sur les modes d'authentification EKS.

Exécutez la commande

Utilisez la commande suivante pour créer le cluster :

```
aws eks create-cluster \
  --region ${AWS_REGION} \
  --cli-input-json \
  "{
      \"name\": \"${CLUSTER_NAME}\",
      \"version\": \"${K8S_VERSION}\",
      \"roleArn\": \"${CLUSTER_ROLE_ARN}\",
      \"resourcesVpcConfig\": {
        \"subnetIds\": ${SUBNETS_JSON},
        \"endpointPublicAccess\": true,
        \"endpointPrivateAccess\": true
      },
      \"computeConfig\": {
        \"enabled\": true,
        \"nodeRoleArn\":\"${NODE_ROLE_ARN}\",
        \"nodePools\": [\"general-purpose\", \"system\"]
      },
      \"kubernetesNetworkConfig\": {
        \"elasticLoadBalancing\": {
          \"enabled\": true
        }
      },
      \"storageConfig\": {
        \"blockStorage\": {
          \"enabled\": true
        }
      },
      \"accessConfig\": {
        \"authenticationMode\": \"API\"
      }
    }
```

Vérifier l'état du cluster

Étape 1 : vérifier la création du cluster

Exécutez la commande suivante pour vérifier l'état de votre cluster. La création d'un cluster prend généralement environ 15 minutes :

```
aws eks describe-cluster --name "${CLUSTER_NAME}" --output json
```

Étape 2 : Mettre à jour kubeconfig

Une fois que le cluster est prêt, mettez à jour votre fichier kubeconfig local pour permettre `kubectl` de communiquer avec le cluster. Cette configuration utilise la AWS CLI pour l'authentification.

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Étape 3 : vérifier les pools de nœuds

Répertoriez les pools de nœuds de votre cluster à l'aide de la commande suivante :

```
kubectl get nodepools
```

Étapes suivantes

- Découvrez comment déployer un exemple de charge de travail sur votre nouveau cluster EKS Auto Mode.

Créez un cluster en mode automatique EKS avec le AWS Management Console

La création d'un cluster en mode automatique EKS dans le AWS Management Console nécessite moins de configuration que les autres options. EKS s'intègre aux réseaux AWS IAM et VPC pour vous aider à créer les ressources associées à un cluster EKS.

Deux options s'offrent à vous pour créer un cluster dans la console :

- Configuration rapide (avec le mode automatique EKS)
- Configuration personnalisée

Dans cette rubrique, vous allez apprendre à créer un cluster en mode automatique EKS à l'aide de l'option de configuration rapide.

Créez un mode automatique EKS à l'aide de l'option de configuration rapide

Vous devez être connecté AWS Management Console avec des autorisations suffisantes pour gérer les AWS ressources, notamment : les EC2 instances, le EC2 réseau, les clusters EKS et les rôles IAM.

1. Accédez à la console EKS
2. Cliquez sur Créer un cluster
3. Confirmez que l'option de configuration rapide est sélectionnée
4. Déterminez les valeurs suivantes ou utilisez les valeurs par défaut d'un cluster de test.
   - Nom du cluster
   - Version de Kubernetes
5. Sélectionnez le rôle IAM du cluster. Si c'est la première fois que vous créez un cluster en mode automatique EKS, utilisez l'option Créer un rôle recommandé.
   - Vous pouvez éventuellement réutiliser un seul rôle IAM de cluster dans votre AWS compte pour tous les clusters en mode automatique d'EKS.
   - Le rôle IAM du cluster inclut les autorisations requises pour le mode automatique d'EKS afin de gérer les ressources, notamment les EC2 instances, les volumes EBS et les équilibreurs de EC2 charge.
   - L'option Créer un rôle recommandé préremplit tous les champs avec des valeurs recommandées. Sélectionnez Suivant, puis Créer. Le rôle utilisera le `AmazonEKSAutoClusterRole` nom suggéré.
   - Si vous avez récemment créé un nouveau rôle, utilisez l'icône Actualiser pour recharger le menu déroulant de sélection des rôles.
6. Sélectionnez le rôle Node IAM. Si c'est la première fois que vous créez un cluster en mode automatique EKS, utilisez l'option Créer un rôle recommandé.
   - Vous pouvez éventuellement réutiliser un rôle IAM de nœud unique dans votre AWS compte pour tous les clusters EKS Auto Mode.
   - Le rôle Node IAM inclut les autorisations requises pour que les nœuds en mode automatique puissent se connecter au cluster. Le rôle Node IAM doit inclure des autorisations permettant de récupérer des images ECR pour vos conteneurs.
   - L'option Créer un rôle recommandé préremplit tous les champs avec des valeurs recommandées. Sélectionnez Suivant, puis Créer. Le rôle utilisera le `AmazonEKSAutoNodeRole` nom suggéré.

- Si vous avez récemment créé un nouveau rôle, utilisez l'icône Actualiser pour recharger le menu déroulant de sélection des rôles.

7. Sélectionnez le VPC pour votre cluster EKS Auto Mode. Choisissez Create VPC pour créer un nouveau VPC pour EKS, ou choisissez un VPC que vous avez créé précédemment pour EKS.
   - Si vous utilisez la console VPC pour créer un nouveau VPC, il est AWS suggéré de créer au moins une passerelle NAT par zone de disponibilité. Sinon, vous pouvez utiliser toutes les autres valeurs par défaut.
   - Pour plus d'informations et de détails sur les exigences du IPv6 cluster, consultezthe section called "Création d'un VPC".
8. (facultatif) Le mode automatique EKS remplit automatiquement les sous-réseaux privés du VPC sélectionné. Vous pouvez supprimer les sous-réseaux indésirables.
   - EKS sélectionne automatiquement les sous-réseaux privés à partir du VPC conformément aux meilleures pratiques. Vous pouvez éventuellement sélectionner des sous-réseaux supplémentaires dans le VPC, tels que des sous-réseaux publics.
9. (facultatif) Sélectionnez Afficher les paramètres de configuration rapide par défaut pour passer en revue toutes les valeurs de configuration du nouveau cluster. Le tableau indique que certaines valeurs ne sont pas modifiables une fois le cluster créé.
10. Sélectionnez Créer un cluster. Notez que la création du cluster peut prendre quinze minutes.

Étapes suivantes

- Découvrez comment déployer un exemple de charge de travail sur votre cluster EKS Auto Mode

Activer le mode automatique EKS sur les clusters EKS existants

Vous pouvez activer le mode automatique EKS sur les clusters EKS existants.

> **Note**
>
> Le mode automatique d'EKS nécessite la version 1.29 ou supérieure de Kubernetes. Le mode automatique EKS n'est pas disponible dans les `mx-central-1` AWS régions `ap-southeast-7` OR.

AWS prend en charge les migrations suivantes :

- Migration de Karpenter vers les nœuds du mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Migrer depuis Karpenter".
- Migration des groupes de nœuds gérés par EKS vers des nœuds en mode automatique d'EKS. Pour de plus amples informations, veuillez consulter the section called "Migrer depuis MNGs".
- Migration d'EKS Fargate vers le mode EKS Auto. Pour de plus amples informations, veuillez consulter the section called "Migrer depuis Fargate".

AWS ne prend pas en charge les migrations suivantes :

- Migration de volumes depuis le contrôleur EBS CSI (à l'aide du module complémentaire Amazon EKS) vers le contrôleur EBS CSI en mode automatique EKS (géré par EKS Auto Mode). PVCs créés avec l'un ne peuvent pas être montés par l'autre, car ils utilisent deux approvisionneurs de volumes Kubernetes différents.
  - Le `eks-auto-mode-ebs-migration-tool`(projet AWS Labs) permet la migration entre le standard EBS CSI StorageClass (`ebs.csi.aws.com`) et le EKS Auto EBS StorageClass CSI (). `ebs.csi.eks.amazonaws.com` Notez que la migration nécessite de supprimer et de recréer PersistentVolumeClaim/PersistentVolume des ressources existantes. La validation dans un environnement hors production est donc essentielle avant la mise en œuvre.
- Migration des équilibreurs de charge du AWS Load Balancer Controller vers le mode automatique EKS

  Vous pouvez installer le AWS Load Balancer Controller sur un cluster Amazon EKS Auto Mode. Utilisez les `loadBalancerClass` options `IngressClass` ou pour associer les ressources de service et d'entrée au Load Balancer Controller ou au mode automatique EKS.
- Migration de clusters EKS avec des configurations réseau alternatives CNIs ou non prises en charge

Référence en matière de migration

Utilisez la référence de migration suivante pour configurer les ressources Kubernetes de manière à ce qu'elles soient détenues par des contrôleurs autogérés ou par le mode automatique EKS.

Capacité	Ressource	Champ	Autogéré	Mode automatique EKS
Stockage en mode bloc	`StorageClass`	`provisioner`	`ebs.csi.aws.com`	`ebs.csi.eks.amazonaws.com`
Equilibrage de charge	`Service`	`loadBalancerClass`	`service.k8s.aws/nlb`	`eks.amazonaws.com/nlb`
Equilibrage de charge	`IngressClass`	`controller`	`ingress.k8s.aws/alb`	`eks.amazonaws.com/alb`
Equilibrage de charge	`IngressClassParams`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Equilibrage de charge	`TargetGroupBinding`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Calcul	`NodeClass`	`apiVersion`	`karpenter.sh/v1`	`eks.amazonaws.com/v1`

Migration de volumes EBS

Lorsque vous migrez des charges de travail vers le mode automatique EKS, vous devez gérer la migration des volumes EBS en raison des différents fournisseurs de pilotes CSI :

- Provisionneur en mode automatique EKS : `ebs.csi.eks.amazonaws.com`
- Provisionneur EBS CSI open source : `ebs.csi.aws.com`

Pour migrer vos volumes persistants, procédez comme suit :

1. Modifier la politique de rétention des volumes : modifiez les versions de plate-forme existantes (PV) `persistentVolumeReclaimPolicy` pour vous `Retain` assurer que le volume EBS sous-jacent n'est pas supprimé.
2. Supprimer le PV de Kubernetes : supprimez l'ancienne ressource PV tout en préservant le volume EBS réel.
3. Créez un nouveau PV avec provisionnement statique : créez un nouveau PV qui fait référence au même volume EBS mais fonctionne avec le pilote CSI cible.
4. Liaison à un nouveau PVC : créez un nouveau PVC qui fait spécifiquement référence à votre PV en utilisant le `volumeName` champ.

Considérations

- Assurez-vous que vos applications sont arrêtées avant de commencer cette migration.
- Sauvegardez vos données avant de commencer le processus de migration.
- Ce processus doit être effectué pour chaque volume persistant.
- La charge de travail doit être mise à jour pour utiliser le nouveau PVC.

Migration des équilibreurs de charge

Vous ne pouvez pas transférer directement les équilibreurs de charge existants du contrôleur d'équilibrage de AWS charge autogéré vers le mode automatique EKS. Vous devez plutôt mettre en œuvre une stratégie de déploiement bleu-vert. Cela implique de conserver la configuration de votre équilibreur de charge existant tout en créant de nouveaux équilibreurs de charge sous le contrôleur géré.

Pour minimiser les interruptions de service, nous recommandons une approche de transfert de trafic basée sur le DNS. Tout d'abord, créez de nouveaux équilibreurs de charge en utilisant le mode automatique EKS tout en maintenant votre configuration existante opérationnelle. Utilisez ensuite le routage DNS (tel que Route 53) pour transférer progressivement le trafic des anciens équilibreurs de charge vers les nouveaux. Une fois que le trafic a été correctement migré et que vous avez vérifié la nouvelle configuration, vous pouvez mettre hors service les anciens équilibreurs de charge et le contrôleur autogéré.

Activer le mode automatique EKS sur un cluster existant

Cette rubrique explique comment activer le mode automatique Amazon EKS sur vos clusters Amazon EKS existants. L'activation du mode automatique sur un cluster existant nécessite la mise à jour des autorisations IAM et la configuration des principaux paramètres du mode automatique EKS. Une fois activé, vous pouvez commencer à migrer vos charges de travail de calcul existantes pour tirer parti des opérations simplifiées et de la gestion automatisée de l'infrastructure du mode automatique.

> **Important**
>
> Vérifiez que la version minimale requise de certains modules complémentaires Amazon EKS est installée avant d'activer le mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Versions complémentaires requises".

Avant de commencer, assurez-vous de disposer d'un accès administrateur à votre cluster Amazon EKS et d'autorisations pour modifier les rôles IAM. Les étapes décrites dans cette rubrique vous expliquent comment activer le mode automatique à l'aide de la AWS CLI AWS Management Console ou de la CLI.

AWS Management Console

Vous devez être connecté à la AWS console avec l'autorisation de gérer IAM, EKS et les EC2 ressources.

> **Note**
>
> Le rôle IAM d'un cluster EKS ne peut pas être modifié une fois le cluster créé. Le mode automatique EKS nécessite des autorisations supplémentaires pour ce rôle. Vous devez associer des politiques supplémentaires au rôle actuel.

Mettre à jour le rôle IAM du cluster

1. Ouvrez la page de présentation de votre cluster dans le AWS Management Console.
2. Sous ARN du rôle IAM du cluster, sélectionnez Afficher dans IAM.
3. Dans le menu déroulant Ajouter des autorisations, sélectionnez Joindre des politiques.

4. Utilisez le champ de recherche pour rechercher et sélectionner les politiques suivantes :
   - `AmazonEKSComputePolicy`
   - `AmazonEKSBlockStoragePolicy`
   - `AmazonEKSLoadBalancingPolicy`
   - `AmazonEKSNetworkingPolicy`
   - `AmazonEKSClusterPolicy`
5. Sélectionnez Ajouter des autorisations
6. Dans l'onglet Relations de confiance, sélectionnez Modifier la politique de confiance
7. Insérez la politique de confiance du rôle IAM du cluster suivante, puis sélectionnez Mettre à jour la politique

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Activer le mode automatique EKS

1. Ouvrez la page de présentation de votre cluster dans le AWS Management Console.
2. Sous EKS Auto Mode, sélectionnez Gérer
3. Activez le mode automatique EKS.
4. Dans le menu déroulant EKS Node Pool, sélectionnez les pools de nœuds par défaut que vous souhaitez créer.
   - En savoir plus sur les pools de nœuds en mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called “Créer un pool de nœuds”.

5. Si vous avez déjà créé un rôle IAM de nœud en mode automatique EKS sur ce AWS compte, sélectionnez-le dans le menu déroulant Rôle de nœud IAM. Si vous n'avez jamais créé ce rôle auparavant, sélectionnez Créer un rôle recommandé et suivez les étapes.

AWS CLI

Prérequis

- Le rôle IAM du cluster EKS existant doit inclure des autorisations suffisantes pour le mode automatique EKS, telles que les politiques suivantes :
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`
- Le rôle IAM du cluster doit disposer d'une politique de confiance mise à jour, y compris l'`sts:TagSession`action. Pour plus d'informations sur la création d'un rôle IAM de cluster, consultezthe section called “ AWS CLI”.
- `aws`CLI installée, connectée et version suffisante. Vous devez être autorisé à gérer IAM, EKS et les EC2 ressources. Pour de plus amples informations, veuillez consulter Configuration.

Procédure

Utilisez les commandes suivantes pour activer le mode automatique EKS sur un cluster existant.

> **Note**
>
> Les fonctionnalités de calcul, de stockage par blocs et d'équilibrage de charge doivent toutes être activées ou désactivées dans la même demande.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=true \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": true}}' \
 --storage-config '{"blockStorage":{"enabled": true}}'
```

Versions complémentaires requises

Si vous prévoyez d'activer le mode automatique EKS sur un cluster existant, vous devrez peut-être mettre à jour certains modules complémentaires. Remarques :

- Cela s'applique uniquement aux clusters existants qui passent en mode automatique EKS.
- Les nouveaux clusters créés avec le mode automatique EKS activé ne nécessitent pas ces mises à jour.

Si l'un des modules complémentaires suivants est installé, assurez-vous qu'il possède au moins la version minimale spécifiée :

Nom du module complémentaire	Version minimale requise
Plugin CNI Amazon VPC pour Kubernetes	v1.19.0-eksbuild.1
Proxy Kube	• v1.26.15-eksbuild.19 • v1.27.16-eksbuild.14 • v1.28.15-eksbuild.4 • v1.29.10-eksbuild.3 • v1.30.6-eksbuild.3 • v1.31.2-eksbuild.3
Pilote CSI Amazon EBS	v1.37.0-eksbuild.1
Contrôleur d'instantané CSI	v8.1.0-eksbuild.2
Agent d'identité du pod EKS	v1.3.4-eksbuild.1

Pour de plus amples informations, veuillez consulter the section called “Mettre à jour un module complémentaire”.

Étapes suivantes

- Pour migrer les charges de travail de gestion des groupes de nœuds, consultezthe section called “Migrer depuis MNGs”.

- Pour migrer depuis Self-Managed Karpenter, consultez. the section called "Migrer depuis Karpenter"

Migrer de Karpenter vers le mode automatique EKS à l'aide de kubectl

Cette rubrique explique le processus de migration des charges de travail de Karpenter vers le mode automatique d'Amazon EKS à l'aide de kubectl. La migration peut être effectuée progressivement, ce qui vous permet de déplacer les charges de travail à votre propre rythme tout en préservant la stabilité du cluster et la disponibilité des applications tout au long de la transition.

L' step-by-stepapproche décrite ci-dessous vous permet d'exécuter Karpenter et le mode automatique d'EKS côte à côte pendant la période de migration. Cette stratégie à double fonctionnement permet de garantir une transition fluide en vous permettant de valider le comportement de la charge de travail en mode automatique d'EKS avant de mettre complètement Karpenter hors service. Vous pouvez migrer les applications individuellement ou en groupe, ce qui vous permet de bénéficier de la flexibilité nécessaire pour répondre à vos exigences opérationnelles spécifiques et à votre tolérance au risque.

Prérequis

Avant de commencer la migration, assurez-vous d'avoir :

- Karpenter v1.1 ou version ultérieure installé sur votre cluster. Pour plus d'informations, consultez la section Mise à niveau vers la version 1.1.0+ dans la documentation de Karpenter.
- `kubectl`installé et connecté à votre cluster. Pour de plus amples informations, veuillez consulter Configuration.

Cette rubrique suppose que vous connaissez Karpenter et NodePools. Pour plus d'informations, consultez la documentation de Karpenter.

Étape 1 : activer le mode automatique EKS sur le cluster

Activez le mode automatique EKS sur votre cluster existant à l'aide de la AWS CLI ou de la console de gestion. Pour de plus amples informations, veuillez consulter the section called "Activer sur le cluster".

> **Note**
>
> Lorsque vous activez le mode automatique EKS, n'activez pas le `general purpose` pool de nœuds à ce stade pendant la transition. Ce pool de nœuds n'est pas sélectif.
> Pour de plus amples informations, veuillez consulter the section called "Passez en revue les pools de nœuds intégrés".

Étape 2 : Création d'un mode automatique EKS altéré NodePool

Créez un nouveau mode automatique NodePool pour EKS avec une teinte. Cela garantit que les pods existants ne seront pas automatiquement programmés sur les nouveaux nœuds du mode automatique EKS. Ce pool de nœuds utilise le mode automatique `default NodeClass` intégré à EKS. Pour de plus amples informations, veuillez consulter the section called "Créer une classe de nœuds".

Exemple de pool de nœuds teinté :

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      taints:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
```

Mettez à jour les exigences relatives au pool de nœuds pour qu'il corresponde à la configuration Karpenter à partir de laquelle vous effectuez la migration. Vous avez besoin d'au moins une exigence.

Étape 3 : Mettre à jour les charges de travail pour la migration

Identifiez et mettez à jour les charges de travail que vous souhaitez migrer vers le mode automatique EKS. Ajoutez à la fois des tolérances et des sélecteurs de nœuds à ces charges de travail :

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

Cette modification permet de planifier la charge de travail sur les nouveaux nœuds du mode automatique EKS.

Le mode automatique EKS utilise des étiquettes différentes de celles de Karpenter. Les étiquettes associées aux instances EC2 gérées commencent par`eks.amazonaws.com`. Pour de plus amples informations, veuillez consulter the section called "Créer un pool de nœuds".

Étape 4 : migrer progressivement les charges de travail

Répétez l'étape 3 pour chaque charge de travail que vous souhaitez migrer. Cela vous permet de déplacer les charges de travail individuellement ou en groupe, en fonction de vos besoins et de votre tolérance au risque.

Étape 5 : Retirez le Karpenter d'origine NodePool

Une fois que toutes les charges de travail ont été migrées, vous pouvez supprimer le Karpenter d'origine : NodePool

```
kubectl delete nodepool <original-nodepool-name>
```

Étape 6 : Supprimer les taches du mode automatique EKS NodePool (facultatif)

Si vous souhaitez que le mode automatique EKS devienne le mode par défaut pour les nouvelles charges de travail, vous pouvez supprimer les traces laissées par le mode automatique EKS : NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      # Remove the taints section
```

Étape 7 : supprimer les sélecteurs de nœuds des charges de travail (facultatif)

Si vous avez supprimé l'altération du mode automatique EKS NodePool, vous pouvez éventuellement supprimer les sélecteurs de nœuds de vos charges de travail, le mode automatique EKS étant désormais le mode par défaut :

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      # Remove the nodeSelector section
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
```

Étape 8 : Désinstaller Karpenter de votre cluster

Les étapes pour supprimer Karpenter dépendent de la façon dont vous l'avez installé. Pour plus d'informations, consultez les instructions d'installation de Karpenter.

Migrer des groupes de nœuds gérés par EKS vers le mode automatique EKS

Lors de la transition de votre cluster Amazon EKS vers le mode automatique EKS, vous pouvez migrer en douceur vos charges de travail existantes à partir de groupes de nœuds gérés (MNGs) à l'aide de l'outil CLI eksctl. Ce processus garantit la disponibilité continue des applications tandis que

le mode automatique d'EKS optimise vos ressources informatiques. La migration peut être effectuée en perturbant le moins possible vos applications en cours d'exécution.

Cette rubrique explique les étapes à suivre pour vider en toute sécurité les pods de vos groupes de nœuds gérés existants et autoriser le mode automatique d'EKS à les replanifier sur les instances nouvellement configurées. En suivant cette procédure, vous pouvez tirer parti de la consolidation intelligente de la charge de travail du mode EKS auto tout en préservant la disponibilité de votre application tout au long de la migration.

Prérequis

- Cluster avec le mode automatique EKS activé
- `eksctl`CLI installée et connectée à votre cluster. Pour de plus amples informations, veuillez consulter Configuration.
- Karpenter n'est pas installé sur le cluster.

Procédure

Utilisez la commande `eksctl` CLI suivante pour lancer le drainage des pods à partir des instances de groupes de nœuds gérés existantes. Le mode automatique d'EKS créera de nouveaux nœuds pour sauvegarder les pods déplacés.

```
eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>
```

Vous devez exécuter cette commande pour chaque groupe de nœuds gérés de votre cluster.

Pour plus d'informations sur cette commande, consultez Supprimer et vider des groupes de nœuds dans la documentation eksctl.

Migrer d'EKS Fargate vers le mode automatique d'EKS

Cette rubrique explique le processus de migration des charges de travail d'EKS Fargate vers le mode automatique d'Amazon EKS à l'aide de. `kubectl` La migration peut être effectuée progressivement, ce qui vous permet de déplacer les charges de travail à votre propre rythme tout en préservant la stabilité du cluster et la disponibilité des applications tout au long de la transition.

L' step-by-stepapproche décrite ci-dessous vous permet d'exécuter EKS Fargate et EKS Auto Mode côte à côte pendant la période de migration. Cette stratégie à double fonctionnement permet de garantir une transition fluide en vous permettant de valider le comportement de la charge de

travail en mode automatique d'EKS avant de mettre complètement EKS Fargate hors service. Vous pouvez migrer les applications individuellement ou en groupe, ce qui vous permet de bénéficier de la flexibilité nécessaire pour répondre à vos exigences opérationnelles spécifiques et à votre tolérance au risque.

Comparaison entre le mode automatique d'Amazon EKS et EKS avec AWS Fargate

Amazon EKS with AWS Fargate reste une option pour les clients qui souhaitent exécuter EKS, mais le mode automatique Amazon EKS est l'approche recommandée pour l'avenir. Le mode automatique d'EKS est entièrement conforme à Kubernetes et prend en charge toutes les primitives Kubernetes en amont et les outils de plateforme tels qu'Istio, que Fargate n'est pas en mesure de prendre en charge. Le mode automatique d'EKS prend également entièrement en charge toutes les options EC2 d'achat d'exécution, y compris les instances GPU et Spot, ce qui permet aux clients de bénéficier de EC2 remises négociées et d'autres mécanismes d'économie. Ces fonctionnalités ne sont pas disponibles lors de l'utilisation d'EKS avec Fargate.

En outre, le mode automatique d'EKS permet aux clients d'obtenir le même modèle d'isolation que Fargate, en utilisant les fonctionnalités de planification standard de Kubernetes pour garantir que EC2 chaque instance exécute un seul conteneur d'applications. En adoptant le mode automatique d'Amazon EKS, les clients peuvent profiter de tous les avantages liés à l'exécution de Kubernetes sur AWS une plateforme entièrement conforme à Kubernetes qui offre la flexibilité nécessaire pour tirer parti de l'ensemble des options d'achat tout en conservant la facilité d'utilisation EC2 et l'abstraction de la gestion de l'infrastructure proposées par Fargate.

Prérequis

Avant de commencer la migration, assurez-vous d'avoir

- Configurez un cluster avec Fargate. Pour de plus amples informations, veuillez consulter the section called "Mise en route".
- Installé et connecté `kubectl` à votre cluster. Pour de plus amples informations, veuillez consulter Configuration.

Étape 1 : Vérifiez le cluster Fargate

1. Vérifiez si le cluster EKS avec Fargate est en cours d'exécution :

```
kubectl get node
```

```
NAME STATUS ROLES AGE VERSION
fargate-ip-192-168-92-52.ec2.internal Ready <none> 25m v1.30.8-eks-2d5f260
fargate-ip-192-168-98-196.ec2.internal Ready <none> 24m v1.30.8-eks-2d5f260
```

2. Vérifiez les pods en cours d'exécution :

```
kubectl get pod -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-6659cb98f6-gxpjz 1/1 Running 0 26m
kube-system coredns-6659cb98f6-gzzsx 1/1 Running 0 26m
```

3. Créez un déploiement dans un fichier appelé `deployment_fargate.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: fargate
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

4. Appliquez le déploiement :

```
kubectl apply -f deployment_fargate.yaml
```

```
deployment.apps/nginx-deployment created
```

5. Vérifiez les pods et les déploiements :

```
kubectl get pod,deploy
```

```
NAME                                     READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5c7479459b-6trtm    1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-g8ssb    1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-mq4mf    1/1     Running   0          61s

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment     3/3     3            3           61s
```

6. Vérifiez le nœud :

```
kubectl get node -owide
```

```
NAME                                     STATUS   ROLES   AGE VERSION
   INTERNAL-IP     EXTERNAL-IP OS-IMAGE       KERNEL-VERSION
 CONTAINER-RUNTIME
fargate-ip-192-168-111-43.ec2.internal   Ready    <none> 31s v1.30.8-eks-2d5f260
 192.168.111.43   <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-117-130.ec2.internal Ready    <none> 36s v1.30.8-eks-2d5f260
 192.168.117.130 <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-74-140.ec2.internal   Ready    <none> 36s v1.30.8-eks-2d5f260
 192.168.74.140   <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
```

Étape 2 : activer le mode automatique EKS sur le cluster

1. Activez le mode automatique EKS sur votre cluster existant à l'aide de la AWS CLI ou de la console de gestion. Pour de plus amples informations, veuillez consulter the section called “Activer sur le cluster”.
2. Vérifiez le pool de nœuds :

```
kubectl get nodepool
```

```
NAME              NODECLASS   NODES   READY   AGE
general-purpose   default     1       True    6m58s
system            default     0       True    3d14h
```

Étape 3 : Mettre à jour les charges de travail pour la migration

Identifiez et mettez à jour les charges de travail que vous souhaitez migrer vers le mode automatique EKS.

Pour migrer une charge de travail de Fargate vers le mode automatique EKS, appliquez l'annotation. `eks.amazonaws.com/compute-type: ec2` Cela garantit que la charge de travail ne sera pas planifiée par Fargate, malgré le profil Fargate, et qu'elle sera absorbée par le mode automatique EKS. NodePool Pour de plus amples informations, veuillez consulter the section called "Créer un pool de nœuds".

1. Modifiez vos déploiements (par exemple, le `deployment_fargate.yaml` fichier) pour changer le type de calcul en `ec2` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      containers:
```

```
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

2. Appliquez le déploiement. Cette modification permet de planifier la charge de travail sur les nouveaux nœuds du mode automatique EKS :

```
kubectl apply -f deployment_fargate.yaml
```

3. Vérifiez que le déploiement est en cours d'exécution dans le cluster EKS Auto Mode :

```
kubectl get pod -o wide
```

```
NAME                                READY   STATUS    RESTARTS   AGE     IP
    NODE                  NOMINATED NODE   READINESS GATES
nginx-deployment-97967b68d-ffxxh   1/1     Running   0          3m31s
 192.168.43.240   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-mbcgj   1/1     Running   0          2m37s
 192.168.43.241   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-qpd8x   1/1     Running   0          2m35s
 192.168.43.242   i-0845aafcb51630ffb   <none>           <none>
```

4. Vérifiez qu'aucun nœud Fargate n'est en cours d'exécution et qu'aucun déploiement n'est en cours d'exécution dans les nœuds gérés en mode automatique EKS :

```
kubectl get node -owide
```

```
NAME                STATUS ROLES  AGE   VERSION             INTERNAL-IP     EXTERNAL-
IP OS-IMAGE                                          KERNEL-VERSION CONTAINER-RUNTIME
i-0845aafcb51630ffb Ready  <none> 3m30s v1.30.8-eks-3c20087 192.168.41.125
 3.81.118.95 Bottlerocket (EKS Auto) 2025.3.14 (aws-k8s-1.30) 6.1.129
 containerd://1.7.25+bottlerocket
```

Étape 4 : migrer progressivement les charges de travail

Répétez l'étape 3 pour chaque charge de travail que vous souhaitez migrer. Cela vous permet de déplacer les charges de travail individuellement ou en groupe, en fonction de vos besoins et de votre tolérance au risque.

Étape 5 : Supprimer le profil Fargate d'origine

Une fois que toutes les charges de travail ont été migrées, vous pouvez supprimer le profil d'origine`fargate`. Remplacez *`<fargate profile name>`* par le nom de votre profil Fargate :

```
aws eks delete-fargate-profile --cluster-name eks-fargate-demo-cluster --fargate-
profile-name <fargate profile name>
```

Étape 6 : réduisez CoreDNS

Comme le mode EKS Auto gère CoreDNS, vous pouvez réduire `coredns` le déploiement à 0 :

```
kubectl scale deployment coredns -n kube-system —-replicas=0
```

Exécutez des exemples de charges de travail dans des clusters en mode automatique EKS

Ce chapitre fournit des exemples de déploiement de différents types de charges de travail sur des clusters Amazon EKS exécutés en mode automatique. Les exemples illustrent les principaux modèles de charge de travail, notamment les exemples d'applications, les applications Web à charge équilibrée, les charges de travail dynamiques utilisant le stockage persistant et les charges de travail présentant des exigences spécifiques en matière de placement des nœuds. Chaque exemple inclut des manifestes complets et des instructions de step-by-step déploiement que vous pouvez utiliser comme modèles pour vos propres applications.

Avant de passer aux exemples, assurez-vous qu'un cluster EKS fonctionne en mode automatique et que vous avez installé la AWS CLI et kubectl. Pour de plus amples informations, veuillez consulter Configuration. Les exemples supposent une connaissance de base des concepts de Kubernetes et des commandes kubectl.

Vous pouvez utiliser ces exemples basés sur des cas pour exécuter des charges de travail dans des clusters en mode automatique EKS.

the section called “Déployez une charge de travail excessive”

> Montre comment déployer un exemple de charge de travail sur un cluster en mode automatique EKS à l'aide de `kubectl` commandes.

the section called "Déployer un équilibreur de charge"

Montre comment déployer une version conteneurisée du jeu 2048 sur Amazon EKS.

the section called "Déployez une charge de travail dynamique"

Montre comment déployer un exemple d'application dynamique sur un cluster en mode automatique EKS.

the section called "Déploiement du contrôle"

Montre comment utiliser une annotation pour contrôler si une charge de travail est déployée sur des nœuds gérés par le mode automatique EKS.

Déployez un exemple de charge de travail gonflée sur un cluster Amazon EKS en mode automatique

Dans ce didacticiel, vous allez apprendre à déployer un exemple de charge de travail sur un cluster en mode automatique EKS et à observer comment il provisionne automatiquement les ressources de calcul requises. Vous utiliserez des `kubectl` commandes pour observer le comportement du cluster et voir par vous-même comment le mode automatique simplifie les opérations de Kubernetes sur. AWSÀ la fin de ce didacticiel, vous comprendrez comment le mode automatique d'EKS répond aux déploiements de charges de travail en gérant automatiquement les ressources informatiques sous-jacentes, sans nécessiter de configuration manuelle des groupes de nœuds.

Prérequis

- Un cluster en mode automatique Amazon EKS. Notez le nom et AWS la région du cluster.
- Un principal IAM, tel qu'un utilisateur ou un rôle, disposant d'autorisations suffisantes pour gérer les ressources réseau, informatiques et EKS.
  - Pour plus d'informations, consultez la section Création de rôles et attachement de politiques dans le Guide de l'utilisateur IAM du Guide de l'utilisateur IAM.
- `aws`CLI installée et configurée avec une identité IAM.
- `kubectl`CLI installée et connectée au cluster.
  - Pour de plus amples informations, veuillez consulter Configuration.

Étape 1 : passer en revue les ressources de calcul existantes (facultatif)

Tout d'abord, utilisez `kubectl` pour répertorier les pools de nœuds de votre cluster.

```
kubectl get nodepools
```

Exemple de sortie :

```
general-purpose
```

Dans ce didacticiel, nous allons déployer une charge de travail configurée pour utiliser le pool de `general-purpose` nœuds. Ce pool de nœuds est intégré au mode automatique d'EKS et inclut des valeurs par défaut raisonnables pour les charges de travail générales, telles que les microservices et les applications Web. Vous pouvez créer votre propre pool de nœuds. Pour de plus amples informations, veuillez consulter the section called "Créer un pool de nœuds".

Ensuite, utilisez `kubectl` pour répertorier les nœuds connectés à votre cluster.

```
kubectl get nodes
```

Si vous venez de créer un cluster en mode automatique EKS, vous n'aurez aucun nœud.

Dans ce didacticiel, vous allez déployer un exemple de charge de travail. Si vous n'avez aucun nœud ou si la charge de travail ne peut pas être adaptée aux nœuds existants, le mode automatique EKS provisionnera un nouveau nœud.

Étape 2 : Déployer un exemple d'application sur le cluster

Passez en revue le déploiement Kubernetes suivant et enregistrez-le sous `inflate.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate
  template:
    metadata:
      labels:
        app: inflate
```

```
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
      containers:
        - name: inflate
          image: public.ecr.aws/eks-distro/kubernetes/pause:3.7
          resources:
            requests:
              cpu: 1
          securityContext:
            allowPrivilegeEscalation: false
```

Notez que le `eks.amazonaws.com/compute-type: auto` sélecteur nécessite que la charge de travail soit déployée sur un nœud Amazon EKS Auto Mode.

Appliquez le déploiement à votre cluster.

```
kubectl apply -f inflate.yaml
```

Étape 3 : Regardez les événements Kubernetes

Utilisez la commande suivante pour surveiller les événements Kubernetes, notamment la création d'un nouveau nœud. `ctrl+c`À utiliser pour arrêter de regarder les événements.

```
kubectl get events -w --sort-by '.lastTimestamp'
```

`kubectl`À utiliser pour répertorier à nouveau les nœuds connectés à votre cluster. Notez le nœud nouvellement créé.

```
kubectl get nodes
```

Étape 4 : Afficher les nœuds et les instances dans la AWS console

Vous pouvez afficher les nœuds du mode automatique EKS dans la console EKS et les EC2 instances associées dans la EC2 console.

EC2 Les instances déployées par le mode automatique d'EKS sont soumises à des restrictions. Vous ne pouvez pas exécuter de commandes arbitraires sur les nœuds du mode automatique EKS.

Étape 5 : supprimer le déploiement

`kubectl`À utiliser pour supprimer l'exemple de déploiement

```
kubectl delete -f inflate.yaml
```

Si aucune autre charge de travail n'est déployée sur votre cluster, le nœud créé par le mode automatique EKS sera vide.

Dans la configuration par défaut, le mode automatique EKS détecte les nœuds vides depuis trente secondes et les arrête.

Utilisez `kubectl` la EC2 console pour confirmer que l'instance associée a été supprimée.

Déployer un exemple de charge de travail Load Balancer en mode EKS Auto

Ce guide explique comment déployer une version conteneurisée du jeu 2048 sur Amazon EKS, avec équilibrage de charge et accessibilité à Internet.

Prérequis

- Un cluster en mode automatique EKS
- `kubectl`configuré pour interagir avec votre cluster
- Autorisations IAM appropriées pour créer des ressources ALB

Étape 1 : Création de l'espace de noms

Créez d'abord un espace de noms dédié pour l'application de jeu 2048.

Créez un fichier nommé `01-namespace.yaml`:

```
apiVersion: v1
kind: Namespace
metadata:
```

```
  name: game-2048
```

Appliquez la configuration de l'espace de noms :

```
kubectl apply -f 01-namespace.yaml
```

Étape 2 : Déploiement de l'application

L'application exécute plusieurs répliques du conteneur de jeu 2048.

Créez un fichier nommé `02-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: "0.5"
```

> **Note**
>
> Si vous recevez une erreur lors du chargement de l'image`public.ecr.aws/l6m2t8p7/docker-2048:latest`, vérifiez que votre rôle Node IAM dispose des autorisations

suffisantes pour extraire des images de l'ECR. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM de nœud". De plus, l'`docker-2048`image présentée dans l'exemple est une `x86_64` image et ne s'exécutera pas sur d'autres architectures.

Composants clés :

- Déploie 5 répliques de l'application
- Utilise une image ECR publique
- Demande 0,5 cœur de processeur par pod
- Expose le port 80 au trafic HTTP

Appliquez le déploiement :

```
kubectl apply -f 02-deployment.yaml
```

Étape 3 : Création du service

Le service expose le déploiement au réseau du cluster.

Créez un fichier nommé `03-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app.kubernetes.io/name: app-2048
```

Composants clés :

- Crée un NodePort service
- Met le port 80 en correspondance avec le port 80 du conteneur

- Utilise le sélecteur d'étiquettes pour trouver les capsules

Appliquez le service :

```
kubectl apply -f 03-service.yaml
```

Étape 4 : Configuration de l'équilibrage de charge

Vous allez configurer une entrée pour exposer l'application à Internet.

Créez d'abord le`IngressClass`. Créez un fichier nommé `04-ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  labels:
    app.kubernetes.io/name: LoadBalancerController
  name: alb
spec:
  controller: eks.amazonaws.com/alb
```

> **Note**
>
> Le mode automatique d'EKS nécessite des balises de sous-réseau pour identifier les sous-réseaux publics et privés.
> Si vous avez créé votre cluster avec`eksctl`, vous possédez déjà ces balises.
> Découvrez comment the section called "Sous-réseaux de balises".

Créez ensuite la ressource Ingress. Créez un fichier nommé `05-ingress.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
```

```
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Composants clés :

- Crée un ALB connecté à Internet
- Utilise le type de cible IP pour le routage direct des pods
- Achemine tout le trafic (/) vers le service de jeu

Appliquez les configurations d'entrée :

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

Étape 5 : vérifier le déploiement

1. Vérifiez que tous les pods fonctionnent :

   ```
   kubectl get pods -n game-2048
   ```

2. Vérifiez que le service est créé :

   ```
   kubectl get svc -n game-2048
   ```

3. Obtenez le point de terminaison ALB :

   ```
   kubectl get ingress -n game-2048
   ```

Le champ ADDRESS de la sortie d'entrée affichera votre point de terminaison ALB. Attendez 2 à 3 minutes pour que l'ALB approvisionne et enregistre toutes les cibles.

Étape 6 : Accédez au jeu

Ouvrez votre navigateur Web et accédez à l'URL du point de terminaison ALB à l'étape précédente. Vous devriez voir l'interface du jeu 2048.

Étape 7 : nettoyage

Pour supprimer toutes les ressources créées dans ce didacticiel :

```
kubectl delete namespace game-2048
```

Cela supprimera toutes les ressources de l'espace de noms, y compris les ressources de déploiement, de service et d'entrée.

Ce qui se passe dans les coulisses

1. Le déploiement crée 5 pods exécutant le jeu 2048
2. Le service fournit un accès réseau stable à ces modules.
3. Mode automatique EKS :
   - Crée un Application Load Balancer dans AWS
   - Configure les groupes cibles pour les pods
   - Configure des règles de routage pour diriger le trafic vers le service

Résolution des problèmes

Si le jeu ne se charge pas :

- Assurez-vous que tous les pods fonctionnent : `kubectl get pods -n game-2048`
- Vérifiez l'état d'entrée : `kubectl describe ingress -n game-2048`
- Vérifier les bilans de santé de l'ALB : vérifier l'état du groupe cible dans la console AWS

Déployer un exemple de charge de travail dynamique en mode automatique EKS

Ce didacticiel vous guidera dans le déploiement d'un exemple d'application dynamique sur votre cluster EKS Auto Mode. L'application écrit des horodatages sur un volume persistant, démontrant

ainsi les fonctionnalités de provisionnement automatique et de persistance des volumes EBS d'EKS Auto Mode.

Prérequis

- Un cluster en mode automatique EKS
- La AWS CLI configurée avec les autorisations appropriées
- `kubectl`installé et configuré
  - Pour de plus amples informations, veuillez consulter Configuration.

Étape 1 : Configuration de votre environnement

1. Définissez vos variables d'environnement :

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. Mettez à jour votre kubeconfig :

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Étape 2 : Création de la classe de stockage

`StorageClass`définit la manière dont le mode automatique EKS approvisionnera les volumes EBS.

Le mode automatique EKS ne crée `StorageClass` pas de Vous devez créer un `StorageClass` référencement `ebs.csi.eks.amazonaws.com` pour utiliser la capacité de stockage du mode automatique EKS.

1. Créez un fichier nommé `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
```

```
parameters:
  type: gp3
  encrypted: "true"
```

2. Appliquez le `StorageClass` :

```
kubectl apply -f storage-class.yaml
```

Composants clés :

- `provisioner: ebs.csi.eks.amazonaws.com`- Utilise le mode automatique EKS
- `volumeBindingMode: WaitForFirstConsumer`- Retarde la création du volume jusqu'à ce qu'un pod en ait besoin
- `type: gp3`- Spécifie le type de volume EBS
- `encrypted: "true"`- EBS utilisera la `aws/ebs` clé par défaut pour chiffrer les volumes créés avec cette classe. Cette action est facultative, mais recommandée.
- `storageclass.kubernetes.io/is-default-class: "true"`- Kubernetes utilisera cette classe de stockage par défaut, sauf si vous spécifiez une classe de volume différente dans le cadre d'une réclamation de volume persistant. Soyez prudent lorsque vous définissez cette valeur si vous effectuez une migration depuis un autre contrôleur de stockage. (facultatif)

Étape 3 : Création de la réclamation de volume persistante

Le PVC demande le stockage au`StorageClass`.

1. Créez un fichier nommé `pvc.yaml`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. Appliquez le PVC :

```
kubectl apply -f pvc.yaml
```

Composants clés :

- `accessModes: ReadWriteOnce`- Le volume peut être monté par un nœud à la fois
- `storage: 8Gi`- Demande un volume de 8 GiB
- `storageClassName: auto-ebs-sc`- Références que `StorageClass` nous avons créées

Étape 4 : Déploiement de l'application

Le déploiement exécute un conteneur qui écrit des horodatages sur le volume persistant.

1. Créez un fichier nommé `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate-stateful
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate-stateful
  template:
    metadata:
      labels:
        app: inflate-stateful
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      containers:
        - name: bash
          image: public.ecr.aws/docker/library/bash:4.4
          command: ["/usr/local/bin/bash"]
          args: ["-c", "while true; do echo $(date -u) >> /data/out.txt; sleep 60;
 done"]
          resources:
```

```
            requests:
              cpu: "1"
          volumeMounts:
            - name: persistent-storage
              mountPath: /data
      volumes:
        - name: persistent-storage
          persistentVolumeClaim:
            claimName: auto-ebs-claim
```

2. Appliquez le déploiement :

```
kubectl apply -f deployment.yaml
```

Composants clés :

- Conteneur bash simple qui écrit des horodatages dans un fichier
- Fixe le PVC sur `/data`
- Demande 1 cœur de processeur
- Utilise le sélecteur de nœuds pour les nœuds gérés par EKS

Étape 5 : vérifier la configuration

1. Vérifiez que le pod fonctionne :

```
kubectl get pods -l app=inflate-stateful
```

2. Vérifiez que le PVC est lié :

```
kubectl get pvc auto-ebs-claim
```

3. Vérifiez le volume EBS :

```
# Get the PV name
PV_NAME=$(kubectl get pvc auto-ebs-claim -o jsonpath='{.spec.volumeName}')
# Describe the EBS volume
aws ec2 describe-volumes \
  --filters Name=tag:CSIVolumeName,Values=${PV_NAME}
```

4. Vérifiez que les données sont en cours d'écriture :

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

Étape 6 : nettoyage

Exécutez la commande suivante pour supprimer toutes les ressources créées dans ce didacticiel :

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

Ce qui se passe dans les coulisses

1. Le PVC demande à être entreposé auprès du `StorageClass`
2. Lorsque le Pod est programmé :
   a. Le mode automatique EKS approvisionne un volume EBS
   b. Crée un PersistentVolume
   c. Attache le volume au nœud
3. Le Pod monte le volume et commence à écrire des horodatages

Contrôleur Snapshot

Le mode automatique EKS est compatible avec le Kubernetes CSI Snapshotter, également connu sous le nom de Snapshot Controller. Cependant, le mode automatique EKS n'inclut pas le contrôleur de capture instantanée. Vous êtes responsable de l'installation et de la configuration du contrôleur de capture instantanée. Pour de plus amples informations, veuillez consulter the section called "Contrôleur d'instantané CSI".

Consultez `VolumeSnapshotClass` ce qui suit qui fait référence à la capacité de stockage du mode automatique EKS.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
```

```
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

En savoir plus sur le Kubernetes CSI Snapshotter.

Configuration des paramètres du mode automatique EKS

Ce chapitre explique comment configurer des aspects spécifiques de vos clusters en mode automatique Amazon Elastic Kubernetes Service (EKS). Bien que le mode automatique d'EKS gère automatiquement la plupart des composants de l'infrastructure, vous pouvez personnaliser certaines fonctionnalités pour répondre aux exigences de votre charge de travail.

À l'aide des options de configuration décrites dans cette rubrique, vous pouvez modifier les paramètres réseau, les ressources de calcul et les comportements d'équilibrage de charge tout en conservant les avantages de la gestion automatisée de l'infrastructure. Avant d'apporter des modifications à la configuration, passez en revue les options disponibles dans les sections suivantes afin de déterminer l'approche la mieux adaptée à vos besoins.

Quelles fonctionnalités souhaitez-vous configurer ?	Option de configuration
Réseau et stockage de nœuds • Configurer le placement des nœuds sur les sous-réseaux publics et privés • Définissez des groupes de sécurité personnalisés pour le contrôle d'accès aux nœuds • Personnalisation des politiques de traduction d'adresses réseau (SNAT) • Activez les politiques réseau Kubernetes, la journalisation et la surveillance détaillées des politiques réseau • Définissez les paramètres de stockage éphémères (taille, IOPS, débit) • Configuration du stockage éphémère chiffré avec des clés KMS personnalisées	the section called “Créer une classe de nœuds”

Quelles fonctionnalités souhaitez-vous configurer ?	Option de configuration
• Isolez le trafic des pods dans des sous-rése aux distincts des nœuds	
Ressources de calcul des nœuds • Sélectionnez des types et des familles d' EC2 instances spécifiques • Définition des architectures de processeur (x86_64,) ARM64 • Configuration des types de capacité (On-Demand, Spot) • Spécifier les zones de disponibilité • Configurer les taches et les étiquettes des nœuds • Définissez le nombre minimum et maximum de nœuds	the section called "Créer un pool de nœuds"
Paramètres de l'Application Load Balancer • Déployez des équilibreurs de charge internes ou connectés à Internet • Configurer la répartition de charge entre zones • Définissez des périodes d'inactivité • Activer HTTP/2 et le support WebSocket • Configuration des paramètres de contrôle de santé • Spécifier les paramètres du certificat TLS • Définition des attributs du groupe cible • Définir le type d'adresse IP (IPv4double pile)	the section called "Créer une classe d'entrée"

Quelles fonctionnalités souhaitez-vous configurer ?	Option de configuration
Paramètres du Network Load Balancer • Configurer le routage IP direct du pod • Activer la répartition de charge entre zones • Définir le délai d'inactivité de la connexion • Configuration des paramètres de contrôle de santé • Spécifier le placement du sous-réseau • Définir le type d'adresse IP (IPv4double pile) • Configurer la préservation de l'adresse IP source du client • Définition des attributs du groupe cible	the section called "Créer un service"
Paramètres de classe de stockage • Définissez les types de volumes EBS (gp3, io1, io2, etc.) • Configuration du chiffrement des volumes et de l'utilisation des clés KMS • Définissez les paramètres d'IOPS et de débit • Définir comme classe de stockage par défaut • Définissez des balises personnalisées pour les volumes provisionnés	the section called "Créez StorageClass"

Quelles fonctionnalités souhaitez-vous configurer ?	Option de configuration
Contrôler l'utilisation de l'ODCR • Configurer le déploiement des charges de travail dans les réservations de capacité EC2 à la demande • Sélectionnez explicitement un ODCRs identifiant spécifique pour une utilisation ciblée de la capacité • Utilisez la sélection basée sur des balises pour cibler les groupes de ODCRs • Filtrer ODCRs par AWS compte propriétaire pour les scénarios entre comptes • Contrôlez si les charges de travail utilisent automatiquement Open ODCRs	the section called "Contrôlez le déploiement de l'ODCR"

Création d'une classe de nœuds pour Amazon EKS

Les classes de nœuds Amazon EKS sont des modèles qui offrent un contrôle précis de la configuration de vos nœuds gérés en mode automatique EKS. Une classe de nœuds définit les paramètres au niveau de l'infrastructure qui s'appliquent aux groupes de nœuds de votre cluster EKS, notamment la configuration réseau, les paramètres de stockage et le balisage des ressources. Cette rubrique explique comment créer et configurer une classe de nœuds pour répondre à vos exigences opérationnelles spécifiques.

Lorsque vous devez personnaliser la façon dont le mode automatique d'EKS approvisionne et configure les EC2 instances au-delà des paramètres par défaut, la création d'une classe de nœuds vous permet de contrôler avec précision les paramètres critiques de l'infrastructure. Par exemple, vous pouvez spécifier l'emplacement des sous-réseaux privés pour améliorer la sécurité, configurer le stockage éphémère des instances pour les charges de travail sensibles aux performances ou appliquer un balisage personnalisé pour la répartition des coûts.

Création d'une classe de nœuds

Pour créer un`NodeClass`, procédez comme suit :

1. Créez un fichier YAML (par exemple,`nodeclass.yaml`) avec votre configuration de classe de nœud
2. Appliquez la configuration à votre cluster à l'aide de `kubectl`
3. Référencez la classe de nœuds dans la configuration de votre pool de nœuds. Pour de plus amples informations, veuillez consulter the section called "Créer un pool de nœuds".

Vous devez `kubectl` être installé et configuré. Pour de plus amples informations, veuillez consulter Configuration.

Exemple de classe de nœuds de base

Voici un exemple de classe de nœud :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: private-compute
spec:
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
  ephemeralStorage:
    size: "160Gi"
```

Cela NodeClass augmente la quantité de stockage éphémère sur le nœud.

Appliquez cette configuration en utilisant :

```
kubectl apply -f nodeclass.yaml
```

Ensuite, faites référence à la classe de nœuds dans la configuration de votre pool de nœuds. Pour de plus amples informations, veuillez consulter the section called "Créer un pool de nœuds".

Créer une entrée d'accès à une classe de nœud

Si vous créez une classe de nœuds personnalisée, vous devez créer une entrée d'accès EKS pour permettre aux nœuds de rejoindre le cluster. EKS crée automatiquement des entrées d'accès lorsque vous utilisez la classe de nœuds et les pools de nœuds intégrés.

Pour plus d'informations sur le fonctionnement des entrées d'accès, consultezthe section called "Accorder des autorisations".

Lorsque vous créez des entrées d'accès pour les classes de nœuds EKS Auto Mode, vous devez utiliser le type d'entrée d'`EC2`accès.

Créer une entrée d'accès avec la CLI

Pour créer une entrée d'accès pour les EC2 nœuds et associer la politique EKS Auto Node :

Mettez à jour les commandes CLI suivantes avec le nom de votre cluster et l'ARN du rôle de nœud. L'ARN du rôle de nœud est spécifié dans la classe de nœud YAML.

```
# Create the access entry for EC2 nodes
aws eks create-access-entry \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --type EC2

# Associate the auto node policy
aws eks associate-access-policy \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy \
  --access-scope type=cluster
```

Créez une entrée d'accès avec CloudFormation

Pour créer une entrée d'accès pour les EC2 nœuds et associer la politique EKS Auto Node :

Mettez à jour ce qui suit CloudFormation avec le nom de votre cluster et l'ARN du rôle de nœud. L'ARN du rôle de nœud est spécifié dans la classe de nœud YAML.

```
EKSAutoNodeRoleAccessEntry:
  Type: AWS::EKS::AccessEntry
  Properties:
    ClusterName: <cluster-name>
```

```
    PrincipalArn: <node-role-arn>
    Type: "EC2"
    AccessPolicies:
      - AccessScope:
          Type: cluster
        PolicyArn: arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy
  DependsOn: [ <cluster-name> ] # previously defined in CloudFormation
```

Pour plus d'informations sur le déploiement CloudFormation de stacks, voir Getting started with CloudFormation

Spécification de la classe de nœuds

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Required fields
  role: MyNodeRole  # IAM role for EC2 instances

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0123456789abcdef0"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
    # Alternative approaches:
    # - id: "sg-0123456789abcdef0"
    # - name: "eks-cluster-security-group"

  # Optional: Pod subnet selector for advanced networking
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0987654321fedcba0"
# must include pod security group selector also
```

```
 podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
    # Alternative using direct security group ID
    # - id: "sg-0123456789abcdef0"

  # Optional: Selects on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        Name: "targeted-odcr"
      # Optional owning account ID filter
      owner: "012345678901"

  # Optional fields
  snatPolicy: Random  # or Disabled

  networkPolicy: DefaultAllow  # or DefaultDeny
  networkPolicyEventLogs: Disabled  # or Enabled

  ephemeralStorage:
    size: "80Gi"    # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000      # Range: 3000-16000
    throughput: 125 # Range: 125-1000
    # Optional KMS key for encryption
    kmsKeyID: "arn:aws: kms:region:account:key/key-id"
    # Accepted formats:
    # KMS Key ID
    # KMS Key ARN
    # Key Alias Name
    # Key Alias ARN

  advancedNetworking:
    # Optional: Controls whether public IP addresses are assigned to instances that are
 launched with the nodeclass.
    # If not set, defaults to the MapPublicIpOnLaunch setting on the subnet.
    associatePublicIPAddress: false

    # Optional: Forward proxy, commonly requires certificateBundles as well
    # for EC2, see https://repost.aws/knowledge-center/eks-http-proxy-containerd-
automation
```

```
    httpsProxy: http://192.0.2.4:3128 #commonly port 3128 (Squid) or 8080 (NGINX) #Max
255 characters
    #httpsProxy: http://[2001:db8::4]:3128 # IPv6 address with port, use []
    noProxy: #Max 50 entries
        - localhost #Max 255 characters each
        - 127.0.0.1
        #- ::1 # IPv6 localhost
        #- 0:0:0:0:0:0:0:1 # IPv6 localhost
        - 169.254.169.254 # EC2 Instance Metadata Service
        #- [fd00:ec2::254] # IPv6 EC2 Instance Metadata Service
        # Domains to exclude, put all VPC endpoints here
        - .internal
        - .eks.amazonaws.com

  # Optional: Custom certificate bundles.
  certificateBundles:
    - name: "custom-cert"
      data: "base64-encoded-cert-data"

  # Optional: Additional EC2 tags (with restrictions)
  tags:
    Environment: "production"
    Team: "platform"
    # Note: Cannot use restricted tags like:
    # - kubernetes.io/cluster/*
    # - karpenter.sh/provisioner-name
    # - karpenter.sh/nodepool
    # - karpenter.sh/nodeclaim
    # - karpenter.sh/managed-by
    # - eks.amazonaws.com/nodeclass
```

Considérations

- Si vous souhaitez vérifier le volume de stockage local d'une instance, vous pouvez décrire le nœud pour voir la ressource de stockage éphémère.
- Chiffrement du volume : EKS utilise la clé KMS personnalisée configurée pour chiffrer le volume racine en lecture seule de l'instance et le read/write volume de données.
- Remplacer le rôle IAM du nœud : si vous modifiez le rôle IAM du nœud associé à un`NodeClass`, vous devrez créer une nouvelle entrée d'accès. EKS crée automatiquement une entrée d'accès pour le rôle IAM du nœud lors de la création du cluster. Le rôle IAM du nœud nécessite la politique

d'accès `AmazonEKSAutoNodePolicy` EKS. Pour de plus amples informations, veuillez consulter the section called "Accorder des autorisations".

- densité maximale de pods - EKS limite le nombre maximum de pods sur un nœud à 110. Cette limite est appliquée après le calcul du nombre maximum de pods existant. Pour de plus amples informations, veuillez consulter the section called "Types d' EC2 instances Amazon".
- Balises - Si vous souhaitez propager des balises depuis Kubernetes vers EC2, vous devez configurer des autorisations IAM supplémentaires. Pour de plus amples informations, veuillez consulter the section called "Identité et accès".
- Classe de nœud par défaut : ne nommez pas votre classe de nœud personnaliséedefault. Cela est dû au fait que le mode automatique d'EKS inclut un `NodeClass` appel `default` qui est automatiquement provisionné lorsque vous activez au moins un appel intégré`NodePool`. Pour plus d'informations sur l'activation de la fonction intégrée`NodePools`, consultezthe section called "Passez en revue les pools de nœuds intégrés".
- **`subnetSelectorTerms`**comportement avec plusieurs sous-réseaux - S'il existe plusieurs sous-réseaux qui répondent aux `subnetSelectorTerms` conditions ou que vous fournissez par ID, le mode automatique EKS crée des nœuds répartis sur les sous-réseaux.
  - Si les sous-réseaux se trouvent dans des zones de disponibilité (AZ) différentes, vous pouvez utiliser les fonctionnalités de Kubernetes telles que les contraintes de propagation de la topologie des pods et le routage adapté à la topologie pour répartir les pods et le trafic entre les zones, respectivement.
  - S'il existe plusieurs sous-réseaux dans la même zone de zone qui correspondent à la`subnetSelectorTerms`, le mode automatique d'EKS crée des pods sur chaque nœud répartis sur les sous-réseaux de cette zone. Le mode automatique EKS crée des interfaces réseau secondaires sur chaque nœud des autres sous-réseaux de la même zone de distribution. Il choisit en fonction du nombre d'adresses IP disponibles dans chaque sous-réseau, afin d'utiliser les sous-réseaux de manière plus efficace. Cependant, vous ne pouvez pas spécifier quel sous-réseau EKS Auto Mode utilise pour chaque pod ; si vous avez besoin de pods pour fonctionner dans des sous-réseaux spécifiques, the section called "Sélection de sous-réseaux pour les pods" utilisez-le plutôt.

Sélection de sous-réseaux pour les pods

Les `podSecurityGroupSelectorTerms` champs `podSubnetSelectorTerms` et permettent des configurations réseau avancées en permettant aux pods de s'exécuter dans des sous-réseaux différents de ceux de leurs nœuds. Cette séparation permet de mieux contrôler le routage du trafic

réseau et les politiques de sécurité. Notez que cela `podSecurityGroupSelectorTerms` est obligatoire avec le`podSubnetSelectorTerms`.

Cas d'utilisation

À utiliser `podSubnetSelectorTerms` lorsque vous devez :

- Séparez le trafic d'infrastructure (node-to-node communication) du trafic d'applications (pod-to-pod communication)
- Appliquez des configurations réseau différentes aux sous-réseaux de nœuds par rapport aux sous-réseaux de pods.
- Mettez en œuvre différentes politiques de sécurité ou règles de routage pour les nœuds et les pods.
- Configurez les proxys inverses ou le filtrage réseau spécifiquement pour le trafic des nœuds sans affecter le trafic des pods. Utilisez `advancedNetworking` et `certificateBundles` pour définir votre proxy inverse et tous les certificats auto-signés ou privés pour le proxy.

Exemple de configuration

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  # Subnets for EC2 instances (nodes)
  subnetSelectorTerms:
    - tags:
        Name: "node-subnet"
        kubernetes.io/role/internal-elb: "1"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  # Separate subnets for pods
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
```

```
        kubernetes.io/role/pod: "1"

  podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
```

Considérations relatives aux sélecteurs de sous-réseaux pour les pods

- Densité de pods réduite : moins de pods peuvent être exécutés sur chaque nœud lors de `podSubnetSelectorTerms` son utilisation, car l'interface réseau principale du nœud se trouve dans le sous-réseau du nœud et ne peut pas être utilisée pour les pods du sous-réseau du pod.
- Limites du sélecteur de sous-réseau : la norme `subnetSelectorTerms` et les `securityGroupSelectorTerms` configurations ne s'appliquent pas à la sélection du sous-réseau du pod.
- Planification du réseau : assurez-vous que l'espace d'adresse IP est suffisant dans les sous-réseaux de nœuds et de pods pour répondre aux exigences de votre charge de travail.
- Configuration du routage : Vérifiez que la table de routage et la liste de contrôle d'accès réseau (ACL) des sous-réseaux du pod sont correctement configurées pour la communication entre le nœud et les sous-réseaux du pod.

Création d'un pool de nœuds pour le mode automatique EKS

Les pools de nœuds Amazon EKS offrent un moyen flexible de gérer les ressources de calcul de votre cluster Kubernetes. Cette rubrique explique comment créer et configurer des pools de nœuds à l'aide de Karpenter, un outil de provisionnement de nœuds qui permet d'optimiser le dimensionnement des clusters et l'utilisation des ressources. Avec les NodePool ressources de Karpenter, vous pouvez définir des exigences spécifiques pour vos ressources de calcul, notamment les types d'instances, les zones de disponibilité, les architectures et les types de capacité.

Vous ne pouvez pas modifier les pools intégrés `system` et les pools de `general-purpose` nœuds. Vous pouvez uniquement les activer ou les désactiver. Pour de plus amples informations, veuillez consulter the section called “Passez en revue les pools de nœuds intégrés”.

La NodePool spécification permet un contrôle précis des ressources de calcul de votre cluster EKS grâce à diverses étiquettes et exigences prises en charge. Il s'agit notamment d'options permettant de spécifier les catégories d' EC2 instances, les configurations du processeur, les zones de disponibilité, les architectures (ARM64/AMD64) et les types de capacité (ponctuels ou à la

demande). Vous pouvez également définir des limites de ressources pour l'utilisation du processeur et de la mémoire, afin de garantir que votre cluster respecte les limites opérationnelles requises.

Le mode automatique d'EKS utilise des étiquettes Kubernetes bien connues pour fournir des méthodes cohérentes et standardisées d'identification des caractéristiques des nœuds. Ces labels, tels que ceux `topology.kubernetes.io/zone` pour les zones de disponibilité et `kubernetes.io/arch` pour l'architecture du processeur, suivent les conventions Kubernetes établies. En outre, les étiquettes spécifiques à EKS (préfixées pare`ks.amazonaws.com/`) étendent cette fonctionnalité avec des attributs AWS spécifiques tels que les types d'instances, les fabricants de processeurs, les capacités du GPU et les spécifications réseau. Ce système d'étiquetage standardisé permet une intégration parfaite avec les outils Kubernetes existants tout en fournissant une intégration approfondie AWS de l'infrastructure.

Créez un NodePool

Suivez ces étapes pour créer un cluster Amazon EKS NodePool pour votre cluster Amazon EKS :

1. Créez un fichier YAML nommé `nodepool.yaml` avec la NodePool configuration requise. Vous pouvez utiliser l'exemple de configuration ci-dessous.
2. Appliquez le NodePool à votre cluster :

```
kubectl apply -f nodepool.yaml
```

3. Vérifiez que le NodePool a été créé avec succès :

```
kubectl get nodepools
```

4. (Facultatif) Surveillez l' NodePool état :

```
kubectl describe nodepool default
```

Assurez-vous que votre NodePool référence est valide NodeClass et existe dans votre cluster. NodeClass définit des AWS configurations spécifiques pour vos ressources de calcul. Pour de plus amples informations, veuillez consulter the section called “Créer une classe de nœuds”.

Exemple NodePool

```
apiVersion: karpenter.sh/v1
```

```
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        billing-team: my-team
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["arm64", "amd64"]

  limits:
    cpu: "1000"
    memory: 1000Gi
```

Étiquettes compatibles avec le mode automatique EKS

Le mode automatique EKS prend en charge les labels bien connus suivants.

> **Note**
>
> Le mode automatique EKS utilise des étiquettes différentes de celles de Karpenter. Les étiquettes associées aux instances EC2 gérées commencent par `eks.amazonaws.com`.

Étiquette	exemple	Description
topology.kubernetes.io/zone	us-east-2a	AWS région
node.kubernetes.io/instance-type	g4dn.8xlarge	AWS type d'instance
kubernetes.io/arch	amd64	Les architectures sont définies par les valeurs GOARCH sur l'instance
karpenter.sh/capacity type	place	Les types de capacité incluent : `spot` `on-demand`
eks.amazonaws.com/instance-hypervisor	nitro	Types d'instances utilisant un hyperviseur spécifique
eks.amazonaws.com/compute-type	auto	Identifie les nœuds gérés en mode automatique EKS
eks.amazonaws.com/ - pris instance-encryption-in-transit en charge	true	Types d'instances prenant en charge (ou non) le chiffrement en transit
eks.amazonaws.com/instance-category	g	Types d'instances de la même catégorie, généralement la chaîne précédant le numéro de génération
eks.amazonaws.com/instance-generation	4	Numéro de génération du type d'instance dans une catégorie d'instance
eks.amazonaws.com/instance-family	g4dn	Types d'instances présentant des propriétés similaires mais des quantités de ressources différentes
eks.amazonaws.com/instance-size	8xlarge	Types d'instances présentant des quantités de ressources similaires mais des propriétés différentes

Étiquette	exemple	Description
eks.amazonaws.com/instance-cpu	32	Numéro de CPUs sur l'instance
eks.amazonaws.com/ instance-cpu-manufacturer	`aws`	Nom du fabricant du processeur
eks.amazonaws.com/instance-memory	131072	Nombre de mégaoctets de mémoire sur l'instance
eks.amazonaws.com/ instance-ebs-bandwidth	9500	Nombre maximal de mégabits d'EBS disponibl es sur l'instance
eks.amazonaws.com/ instance-network-bandwidth	131072	Nombre de mégabits de référence disponibles sur l'instance
eks.amazonaws.com/ instance-gpu-name	t4	Nom du GPU de l'instance, si disponible
eks.amazonaws.com/ instance-gpu-manufacturer	nvidia	Nom du fabricant du GPU
eks.amazonaws.com/ instance-gpu-count	1	Numéro de GPUs sur l'instance
eks.amazonaws.com/ instance-gpu-memory	16384	Nombre de mégaoctets de mémoire sur le GPU
eks.amazonaws.com/ instance-local-nvme	900	Nombre de gibioctets de stockage NVME local sur l'instance

> **Note**
>
> Le mode automatique d'EKS ne prend en charge que certaines instances et impose des exigences de taille minimale. Pour de plus amples informations, veuillez consulter the section called "Référence d'instance prise en charge par EKS Auto Mode".

Étiquettes EKS Auto non prises en charge

Le mode automatique EKS ne prend pas en charge les libellés suivants.

- Le mode automatique d'EKS ne fonctionne que sous Linux
  - `node.kubernetes.io/windows-build`
  - `kubernetes.io/os`

Désactiver les pools de nœuds intégrés

Si vous créez des pools de nœuds personnalisés, vous pouvez désactiver les pools de nœuds intégrés. Pour de plus amples informations, veuillez consulter the section called "Passez en revue les pools de nœuds intégrés".

Cluster sans pools de nœuds intégrés

Vous pouvez créer un cluster sans les pools de nœuds intégrés. Cela est utile lorsque votre organisation a créé des pools de nœuds personnalisés.

> **Note**
>
> Lorsque vous créez un cluster sans pool de nœuds intégré, celui-ci n'`default` NodeClass est pas automatiquement provisionné. Vous devrez créer une personnalisation NodeClass. Pour de plus amples informations, veuillez consulter the section called "Créer une classe de nœuds".

Vue d'ensemble :

1. Créez un cluster EKS avec `nodePools` les deux `nodeRoleArn` valeurs vides.
   - Exemple eksctl `autoModeConfig` :

     ```
     autoModeConfig:
       enabled: true
       nodePools: []
       # Do not set a nodeRoleARN
     ```

     Pour de plus amples informations, consultez the section called "CLI exsctl".

2. Création d'une classe de nœud personnalisée avec un rôle de nœud ARN
   - Pour de plus amples informations, consultez the section called “Créer une classe de nœuds”.
3. Créez une entrée d'accès pour la classe de nœud personnalisée
   - Pour de plus amples informations, consultez the section called “Créer une entrée d'accès à une classe de nœud”.
4. Créez un pool de nœuds personnalisé, comme décrit ci-dessus.

Perturbation

Vous pouvez configurer le mode automatique d'EKS pour perturber les nœuds NodePool de différentes manières. Vous pouvez utiliser `spec.disruption.consolidationPolicyspec.disruption.consolidateAfter`, ou`spec.template.spec.expireAfter`. Vous pouvez également limiter le taux d'interruption du mode automatique EKS par le biais NodePool de`spec.disruption.budgets`. Vous pouvez également contrôler les fenêtres temporelles et le nombre de nœuds interrompus simultanément. Pour obtenir des instructions sur la configuration de ce comportement, consultez la section Interruption dans la documentation de Karpenter.

Vous pouvez configurer les interruptions pour les pools de nœuds afin de :

- Identifiez les cas de sous-utilisation des instances et consolidez les charges de travail.
- Créez un budget d'interruption du pool de nœuds afin de limiter le taux de terminaison des nœuds en raison de la dérive, du vide et de la consolidation.

Par défaut, mode automatique EKS :

- Consolide les instances sous-utilisées.
- Met fin aux instances au bout de 336 heures.
- Définit un budget d'interruption unique de 10 % des nœuds.
- Permet de remplacer les nœuds en cas de dérive lorsqu'une nouvelle AMI en mode automatique est publiée, ce qui se produit environ une fois par semaine.

Période de grâce de résiliation

Lorsque a n'`terminationGracePeriod`est pas explicitement défini sur un EKS Auto NodePool, le système applique automatiquement un délai de grâce de résiliation par défaut de 24 heures

au délai de résiliation associé NodeClaim. Les clients d'EKS Auto ne verront pas de valeur `terminationGracePeriod` par défaut dans leurs NodePool configurations personnalisées, mais ils observeront cette valeur par défaut sur le NodeClaim. La fonctionnalité reste cohérente, que le délai de grâce soit explicitement défini sur le NodePool ou défini par défaut sur le NodeClaim, ce qui garantit un comportement de terminaison des nœuds prévisible dans l'ensemble du cluster.

Créez un IngressClass pour configurer un Application Load Balancer

Le mode automatique d'EKS automatise les tâches de routine pour l'équilibrage de charge, notamment l'exposition des applications du cluster à Internet.

AWS suggère d'utiliser des équilibreurs de charge d'application (ALB) pour gérer le trafic HTTP et HTTPS. Les équilibreurs de charge d'application peuvent acheminer les demandes en fonction de leur contenu. Pour plus d'informations sur les équilibreurs de charge des applications, consultez Qu'est-ce qu'Elastic Load Balancing ?

Le mode automatique EKS crée et configure les équilibreurs de charge d'application ()ALBs. Par exemple, le mode automatique EKS crée un équilibreur de charge lorsque vous créez un objet `Ingress` Kubernetes et le configure pour acheminer le trafic vers la charge de travail de votre cluster.

Présentation

1. Créez une charge de travail que vous souhaitez exposer à Internet.
2. Créez une `IngressClassParams` ressource en spécifiant AWS des valeurs de configuration spécifiques telles que le certificat à utiliser SSL/TLS et les sous-réseaux VPC.
3. Créez une `IngressClass` ressource en spécifiant que le mode automatique EKS sera le contrôleur de la ressource.
4. Créez une `Ingress` ressource qui associe un chemin et un port HTTP à une charge de travail de cluster.

Le mode automatique EKS créera un Application Load Balancer qui pointe vers la charge de travail spécifiée dans la `Ingress` ressource, en utilisant la configuration de l'équilibreur de charge spécifiée dans la ressource. `IngressClassParams`

Prérequis

- Mode automatique EKS activé sur un cluster Amazon EKS

- Kubectl configuré pour se connecter à votre cluster
  - Vous pouvez l'utiliser `kubectl apply -f <filename>` pour appliquer les exemples de fichiers YAML de configuration ci-dessous à votre cluster.

> **Note**
>
> Le mode automatique d'EKS nécessite des balises de sous-réseau pour identifier les sous-réseaux publics et privés.
> Si vous avez créé votre cluster avec`eksctl`, vous possédez déjà ces balises.
> Découvrez comment the section called "Sous-réseaux de balises".

Étape 1 : créer une charge de travail

Créez une charge de travail que vous souhaitez exposer à Internet. Il peut s'agir de n'importe quelle ressource Kubernetes qui gère le trafic HTTP, telle qu'un déploiement ou un service. Pour cet exemple, nous utilisons un service HTTP simple appelé `service-2048` qui sert le trafic HTTP sur le port 80.

Étape 2 : créer IngressClassParams

Créez un `IngressClassParams` objet pour spécifier AWS des options de configuration spécifiques pour l'Application Load Balancer. Dans cet exemple, nous créons une `IngressClassParams` ressource nommée `alb` (que vous utiliserez à l'étape suivante) qui spécifie le schéma d'équilibrage de charge tel que `internet-facing` dans un fichier appelé`alb-ingressclassparams.yaml`.

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

Appliquez la configuration à votre cluster :

```
kubectl apply -f alb-ingressclassparams.yaml
```

Étape 3 : créer IngressClass

Créez un `IngressClass` qui fait référence aux valeurs de configuration AWS spécifiques définies dans la `IngressClassParams` ressource dans un fichier nommé`alb-ingressclass.yaml`. Notez le nom du`IngressClass`. Dans cet exemple, les `IngressClass` et `IngressClassParams` sont tous deux nommés`alb`.

Utilisez l'`is-default-class`annotation pour contrôler si `Ingress` les ressources doivent utiliser cette classe par défaut.

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    # Use this annotation to set an IngressClass as Default
    # If an Ingress doesn't specify a class, it will use the Default
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  # Configures the IngressClass to use EKS Auto Mode
  controller: eks.amazonaws.com/alb
  parameters:
    apiGroup: eks.amazonaws.com
    kind: IngressClassParams
    # Use the name of the IngressClassParams set in the previous step
    name: alb
```

Pour plus d'informations sur les options de configuration, consultezthe section called “IngressClassParams Référence”.

Appliquez la configuration à votre cluster :

```
kubectl apply -f alb-ingressclass.yaml
```

Étape 4 : créer une entrée

Créez une `Ingress` ressource dans un fichier nommé`alb-ingress.yaml`. Le but de cette ressource est d'associer les chemins et les ports de l'Application Load Balancer aux charges de travail de votre cluster. Dans cet exemple, nous créons une `Ingress` ressource nommée `2048-ingress` qui achemine le trafic vers un service nommé `service-2048` sur le port 80.

Pour plus d'informations sur la configuration de cette ressource, consultez Ingress dans la documentation de Kubernetes.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: 2048-ingress
spec:
  # this matches the name of IngressClass.
  # this can be omitted if you have a default ingressClass in cluster: the one with
 ingressclass.kubernetes.io/is-default-class: "true"  annotation
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Appliquez la configuration à votre cluster :

```
kubectl apply -f alb-ingress.yaml
```

Étape 5 : Vérifier le statut

`kubectl`À utiliser pour connaître le statut du`Ingress`. La disponibilité de l'équilibreur de charge peut prendre quelques minutes.

Utilisez le nom de la `Ingress` ressource que vous avez définie à l'étape précédente. Par exemple :

```
kubectl get ingress 2048-ingress
```

Une fois que la ressource est prête, récupérez le nom de domaine de l'équilibreur de charge.

```
kubectl get ingress 2048-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Pour afficher le service dans un navigateur Web, passez en revue le port et le chemin spécifiés dans le `Ingress` sauvetage.

Étape 6 : nettoyage

Pour nettoyer l'équilibreur de charge, utilisez la commande suivante :

```
kubectl delete ingress 2048-ingress
kubectl delete ingressclass alb
kubectl delete ingressclassparams alb
```

Le mode automatique EKS supprimera automatiquement l'équilibreur de charge associé dans votre AWS compte.

IngressClassParams Référence

Le tableau ci-dessous est une référence rapide pour les options de configuration couramment utilisées.

Champ	Description	Exemple de valeur
`scheme`	Définit si l'ALB est interne ou connecté à Internet	`internet-facing`
`namespaceSelector`	Limite les espaces de noms qui peuvent l'utiliser IngressClass	`environment: prod`
`group.name`	Regroupe plusieurs entrées pour partager un seul ALB	`retail-apps`
`ipAddressType`	Définit le type d'adresse IP pour l'ALB	`dualstack`
`subnets.ids`	Liste des sous-réseaux pour le déploiement IDs d'ALB	`subnet-xxxx, subnet-yyyy`
`subnets.tags`	Filtres de balises pour sélectionner des sous-réseaux pour ALB	`Environment: prod`

Champ	Description	Exemple de valeur
`certificateARNs`	ARNs de certificats SSL à utiliser	`arn:aws: acm:region:account:certificate/id`
`tags`	Tags personnalisés pour les AWS ressources	`Environment: prod, Team: platform`
`loadBalancerAttributes`	Attributs spécifiques à l'équilibreur de charge	`idle_timeout.timeout_seconds: 60`

Considérations

- Vous ne pouvez pas utiliser les annotations sur et IngressClass pour configurer les équilibreurs de charge avec le mode automatique EKS.
- Vous devez mettre à jour le rôle IAM du cluster pour permettre la propagation des balises depuis Kubernetes vers les ressources Load AWS Balancer. Pour de plus amples informations, veuillez consulter the section called "AWS Balises personnalisées pour les ressources EKS Auto".
- Pour plus d'informations sur l'association de ressources au mode automatique EKS ou au AWS Load Balancer Controller autogéré, consultez. the section called "Référence en matière de migration"
- Pour plus d'informations sur la résolution des problèmes liés aux équilibreurs de charge, consultezthe section called "Dépannage".
- Pour plus d'informations sur l'utilisation de la fonctionnalité d'équilibrage de charge du mode automatique EKS, voirthe section called "Equilibrage de charge".

Les tableaux suivants fournissent une comparaison détaillée des modifications IngressClassParams, des annotations d'entrée et des TargetGroupBinding configurations pour le mode automatique EKS. Ces tableaux mettent en évidence les principales différences entre la capacité d'équilibrage de charge du mode automatique d'EKS et le contrôleur d'équilibrage de charge open source, notamment les modifications de version de l'API, les fonctionnalités obsolètes et les noms de paramètres mis à jour.

IngressClassParams

Précédent	New	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modification de la version de l'API
`spec.certificateArn`	`spec.certificateARNs`	Support pour plusieurs certificats ARNs
`spec.subnets.tags`	`spec.subnets.matchTags`	Schéma de correspondance des sous-réseaux modifié
`spec.listeners.listenerAttributes`	`spec.listeners.attributes`	Dénomination simplifiée des attributs

Annotations d'entrée

Précédent	New	Description
`kubernetes.io/ingress.class`	Non pris en charge	Utilisation `spec.ingressClassName` sur les objets Ingress
`alb.ingress.kubernetes.io/group.name`	Non pris en charge	Spécifier les groupes dans IngressClass uniquement
`alb.ingress.kubernetes.io/waf-acl-id`	Non pris en charge	Utilisez WAF v2 à la place
`alb.ingress.kubernetes.io/web-acl-id`	Non pris en charge	Utilisez WAF v2 à la place
`alb.ingress.kubernetes.io/shield-advanced-protection`	Non pris en charge	Intégration Shield désactivée

Précédent	New	Description
`alb.ingress.kubernetes.io/auth-type: oidc`	Non pris en charge	Le type d'authentification OIDC n'est actuellement pas pris en charge

TargetGroupBinding

Précédent	New	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modification de la version de l'API
`spec.targetType` optionnel	`spec.targetType` requis	Spécification explicite du type de cible
`spec.networking.ingress.from`	Non pris en charge	Ne prend plus en charge le NLB sans groupes de sécurité

Utiliser les annotations de service pour configurer les équilibreurs de charge réseau

Découvrez comment configurer les équilibreurs de charge réseau (NLB) dans Amazon EKS à l'aide des annotations du service Kubernetes. Cette rubrique décrit les annotations prises en charge par le mode automatique d'EKS pour personnaliser le comportement du NLB, notamment l'accessibilité à Internet, les bilans de santé, la SSL/TLS résiliation et les modes de ciblage IP.

Lorsque vous créez un service Kubernetes de type `LoadBalancer` EKS Auto Mode, EKS approvisionne et configure automatiquement un AWS Network Load Balancer en fonction des annotations que vous spécifiez. Cette approche déclarative vous permet de gérer les configurations d'équilibreurs de charge directement via vos manifestes Kubernetes, en maintenant l'infrastructure en tant que pratiques de code.

Le mode automatique EKS gère le provisionnement du Network Load Balancer par défaut pour tous les types LoadBalancer de services. Aucune installation ou configuration de contrôleur supplémentaire n'est requise. La `loadBalancerClass: eks.amazonaws.com/nlb` spécification

est automatiquement définie comme valeur par défaut du cluster, ce qui rationalise le processus de déploiement tout en maintenant la compatibilité avec les charges de travail Kubernetes existantes.

> **Note**
>
> Le mode automatique d'EKS nécessite des balises de sous-réseau pour identifier les sous-réseaux publics et privés.
> Si vous avez créé votre cluster avec`eksctl`, vous possédez déjà ces balises.
> Découvrez comment the section called "Sous-réseaux de balises".

Service d'échantillons

Pour plus d'informations sur la `Service` ressource Kubernetes, consultez la documentation Kubernetes.

Consultez l'exemple de `Service` ressource ci-dessous :

```
apiVersion: v1
kind: Service
metadata:
  name: echoserver
  annotations:
    # Specify the load balancer scheme as internet-facing to create a public-facing
 Network Load Balancer (NLB)
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  selector:
    app: echoserver
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  # Specify the new load balancer class for NLB as part of EKS Auto Mode feature
  # For clusters with Auto Mode enabled, this field can be omitted as it's the default
  loadBalancerClass: eks.amazonaws.com/nlb
```

Annotations couramment utilisées

Le tableau suivant répertorie les annotations couramment utilisées prises en charge par le mode automatique EKS. Notez qu'il est possible que le mode automatique d'EKS ne prenne pas en charge toutes les annotations.

> **Tip**
>
> Toutes les annotations suivantes doivent être préfixées par `service.beta.kubernetes.io/`

Champ	Description	Exemple
`aws-load-balancer-type`	Spécifie le type d'équilibreur de charge. À utiliser `external` pour les nouveaux déploiements.	`external`
`aws-load-balancer-nlb-target-type`	Spécifie s'il faut acheminer le trafic vers les instances de nœuds ou directement vers le pod IPs. `instance`À utiliser pour les déploiements standard ou `ip` pour le routage direct des pods.	`instance`
`aws-load-balancer-scheme`	Contrôle si l'équilibreur de charge est interne ou connecté à Internet.	`internet-facing`
`aws-load-balancer-healthcheck-protocol`	Protocole de contrôle de santé pour le groupe cible. Les options courantes sont `TCP` (par défaut) ou`HTTP`.	`HTTP`
`aws-load-balancer-healthcheck-path`	Le chemin HTTP pour les contrôles de santé lors de	`/healthz`

Champ	Description	Exemple
	l'utilisation HTTP/HTTPS du protocole.	
`aws-load-balancer-healthcheck-port`	Port utilisé pour les contrôles de santé. Il peut s'agir d'un numéro de port spécifique `outraffic-port` .	`traffic-port`
`aws-load-balancer-subnets`	Spécifie les sous-réseaux dans lesquels créer l'équilibreur de charge. Peut utiliser un sous-réseau IDs ou des noms.	`subnet-xxxx, subnet-yyyy`
`aws-load-balancer-ssl-cert`	ARN du certificat SSL fourni par Certificate Manager for HTTPS/TLS. AWS	`arn:aws: acm:region:account:certificate/cert-id`
`aws-load-balancer-ssl-ports`	Spécifie les ports qui doivent utiliser le protocole SSL/TLS.	`443, 8443`
`load-balancer-source-ranges`	Les plages CIDR sont autorisées à accéder à l'équilibreur de charge.	`10.0.0.0/24, 192.168.1.0/24`
`aws-load-balancer-additional-resource-tags`	AWS Balises supplémentaires à appliquer à l'équilibreur de charge et aux ressources associées.	`Environment=prod,Team=platform`
`aws-load-balancer-ip-address-type`	Spécifie si l'équilibreur de charge utilise IPv4 ou double pile (IPv4 + IPv6).	`ipv4` ou `dualstack`

Considérations

- Vous devez mettre à jour le rôle IAM du cluster pour permettre la propagation des balises depuis Kubernetes vers les ressources Load AWS Balancer. Pour de plus amples informations, veuillez consulter the section called "AWS Balises personnalisées pour les ressources EKS Auto".
- Pour plus d'informations sur l'association de ressources au mode automatique EKS ou au AWS Load Balancer Controller autogéré, consultez. the section called "Référence en matière de migration"
- Pour plus d'informations sur la résolution des problèmes liés aux équilibreurs de charge, consultezthe section called "Dépannage".
- Pour plus d'informations sur l'utilisation de la fonctionnalité d'équilibrage de charge du mode automatique EKS, voirthe section called "Equilibrage de charge".

Lors de la migration vers le mode automatique EKS pour l'équilibrage de charge, plusieurs modifications des annotations de service et des configurations des ressources sont nécessaires. Les tableaux suivants présentent les principales différences entre les implémentations précédentes et nouvelles, notamment les options non prises en charge et les alternatives recommandées.

Annotations de service

Précédent	New	Description
`service.beta.kubernetes.io/load-balancer-source-ranges`	Non pris en charge	Utilisation `spec.loadBalancerSourceRanges` sur le service
`service.beta.kubernetes.io/aws-load-balancer-type`	Non pris en charge	Utilisation `spec.loadBalancerClass` sur le service
`service.beta.kubernetes.io/aws-load-balancer-internal`	Non pris en charge	Utiliser `service.beta.kubernetes.io/aws-load-balancer-scheme`
Différents attributs de l'équilibreur de charge	Non pris en charge	Utiliser `service.beta.kubernetes.io/`

Précédent	New	Description
		`aws-load-balancer-attributes`
`service.beta.kubernetes.io/aws-load-balancer-proxy-protocol`	Non pris en charge	Utiliser à la `service.beta.kubernetes.io/aws-load-balancer-attributes` place
`service.beta.kubernetes.io/aws-load-balancer-access-log-enabled`	Non pris en charge	Utiliser à la `service.beta.kubernetes.io/aws-load-balancer-attributes` place
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name`	Non pris en charge	Utiliser à la `service.beta.kubernetes.io/aws-load-balancer-attributes` place
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix`	Non pris en charge	Utiliser à la `service.beta.kubernetes.io/aws-load-balancer-attributes` place
`service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled`	Non pris en charge	Utiliser à la `service.beta.kubernetes.io/aws-load-balancer-attributes` place

Pour effectuer une migration à partir d'annotations d'attributs d'équilibreur de charge obsolètes, consolidez ces paramètres dans l'annotation. `service.beta.kubernetes.io/aws-load-balancer-attributes` Cette annotation accepte une liste séparée par des virgules de paires clé-valeur pour les différents attributs de l'équilibreur de charge. Par exemple, pour spécifier le protocole proxy, la journalisation des accès et l'équilibrage de charge entre zones, utilisez le format suivant :

```
service.beta.kubernetes.io/aws-load-balancer-attributes:
 access_logs.s3.enabled=true,access_logs.s3.bucket=my-bucket,access_logs.s3.prefix=my-
prefix,load_balancing.cross_zone.enabled=true
```

Ce format consolidé fournit un moyen plus cohérent et flexible de configurer les attributs de l'équilibreur de charge tout en réduisant le nombre d'annotations individuelles nécessaires. Passez en revue vos configurations de service existantes et mettez-les à jour pour utiliser ce format consolidé.

TargetGroupBinding

Précédent	New	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modification de la version de l'API
`spec.targetType` optionnel	`spec.targetType` requis	Spécification explicite du type de cible
`spec.networking.ingress.from`	Non pris en charge	Ne prend plus en charge le NLB sans groupes de sécurité

Création d'une classe de stockage

`StorageClass`Dans le mode automatique d'Amazon EKS, A définit la manière dont les volumes Amazon EBS sont automatiquement provisionnés lorsque les applications demandent un stockage persistant. Cette page explique comment créer et configurer un `StorageClass` qui fonctionne avec le mode automatique Amazon EKS pour approvisionner des volumes EBS.

En configurant un`StorageClass`, vous pouvez définir les paramètres par défaut pour vos volumes EBS, notamment le type de volume, le chiffrement, les IOPS et d'autres paramètres de stockage. Vous pouvez également configurer le `StorageClass` pour utiliser les clés AWS KMS pour la gestion du chiffrement.

Le mode automatique EKS ne crée `StorageClass` pas de Vous devez créer un `StorageClass` référencement `ebs.csi.eks.amazonaws.com` pour utiliser la capacité de stockage du mode automatique EKS.

Créez d'abord un fichier nommé `storage-class.yaml` :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

Ensuite, appliquez la classe de stockage à votre cluster.

```
kubectl apply -f storage-class.yaml
```

Composants clés :

- `provisioner: ebs.csi.eks.amazonaws.com`- Utilise le mode automatique EKS
- `volumeBindingMode: WaitForFirstConsumer`- Retarde la création du volume jusqu'à ce qu'un pod en ait besoin
- `type: gp3`- Spécifie le type de volume EBS
- `encrypted: "true"`- EBS chiffrera tous les volumes créés à l'aide du. `StorageClass` EBS utilisera l'alias de `aws/ebs` clé par défaut. Pour plus d'informations, consultez Comment fonctionne le chiffrement Amazon EBS dans le guide de l'utilisateur Amazon EBS. Cette valeur est facultative mais suggérée.
- `storageclass.kubernetes.io/is-default-class: "true"`- Kubernetes utilisera cette classe de stockage par défaut, sauf si vous spécifiez une classe de volume différente dans le cadre d'une réclamation de volume persistant. Cette valeur est facultative. Soyez prudent lorsque vous définissez cette valeur si vous effectuez une migration depuis un autre contrôleur de stockage.

Utiliser une clé KMS autogérée pour chiffrer les volumes EBS

Pour utiliser une clé KMS autogérée afin de chiffrer des volumes EBS automatisés par le mode automatique EKS, vous devez :

1. Créez une clé KMS autogérée.

- Pour plus d'informations, consultez Création d'une clé KMS de chiffrement symétrique ou Comment Amazon Elastic Block Store (Amazon EBS) utilise KMS dans le guide de l'utilisateur KMS.

2. Créez une nouvelle politique qui autorise l'accès à la clé KMS.
   - Utilisez l'exemple de stratégie IAM ci-dessous pour créer la stratégie. Insérez l'ARN de la nouvelle clé KMS autogérée. Pour plus d'informations, consultez la section Création de rôles et attachement de politiques (console) dans le guide de l'utilisateur AWS IAM.
3. Associez la politique au rôle de cluster EKS.
   - Utilisez la AWS console pour trouver l'ARN du rôle de cluster EKS. Les informations relatives au rôle sont visibles dans la section Vue d'ensemble. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM du cluster".
4. Mettez à jour le `StorageClass` pour faire référence à l'ID de clé KMS `parameters.kmsKeyId` sur le terrain.

Exemple de politique KMS IAM autogérée

Mettez à jour les valeurs suivantes dans la politique ci-dessous :

- `<account-id>`— Votre identifiant de AWS compte, tel que `111122223333`
- `<aws-region>`— La AWS région de votre cluster, telle que `us-west-2`

```
{
  "Version": "2012-10-17",
  "Id": "key-auto-policy-3",
  "Statement": [
      {
          "Sid": "Enable IAM User Permissions",
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws: iam::<account-id>:root"
          },
          "Action": "kms:*",
          "Resource": "*"
      },
      {
        "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
        "Effect": "Allow",
```

```
        "Principal": {
            "AWS": "*"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:CallerAccount": "<account-id>",
                "kms:ViaService": "ec2.<aws-region>.amazonaws.com"
            }
        }
    }
  ]
}
```

Exemple de KMS autogéré **`StorageClass`**

```
parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: <custom-key-arn>
```

`StorageClass`Référence des paramètres

Pour des informations générales sur les `StorageClass` ressources Kubernetes, consultez la section Classes de stockage dans la documentation Kubernetes.

THe `parameters`section de la `StorageClass` ressource est spécifique à AWS. Utilisez le tableau suivant pour passer en revue les options disponibles.

Paramètres	Valeurs	Par défaut	Description
« csi.storage.k8s.io/fstype »	xfs, ext2, ext3, ext4	poste 4	Type de système de fichiers qui

Paramètres	Valeurs	Par défaut	Description
			sera formaté lors de la création du volume. Ce paramètre distingue les majuscules et minuscules !
« type »	io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1	gp3	Type de volume EBS.
« IOPS par RGB »			Opérations d'E/S par seconde et par GiB. Peut être spécifié pour IO1 IO2, et les GP3 volumes.

Paramètres	Valeurs	Par défaut	Description
« Autoriser Auto IOPSPer GBIncreas e »	true, false	false	Lorsque `"true"` le pilote CSI augmente le nombre d'IOPS pour un volume trop faible pour correspon dre à la plage d'IOPS prise en charge par. `iopsPerGB * <volume size>` AWS Cela permet au provisionnement dynamique de toujours réussir, même lorsque l'utilisa teur spécifie une capacité ou une `iopsPerGB` valeur de PVC trop faible. D'autre part, cela peut entraîner des coûts supplémentaires, car ces volumes ont des IOPS plus élevés que ceux demandés dans`iopsPerGB` .
« iops »			Opérations d'E/S par seconde. Peut être spécifié pour IO1 IO2, et les GP3 volumes.

Paramètres	Valeurs	Par défaut	Description
« débit »		125	Débit en Mbits/s. Efficace uniquemen t lorsque le type de volume gp3 est spécifié.
« crypté »	true, false	false	Si le volume doit être chiffré ou non. Les valeurs valides sont « vrai » ou « faux ».
« Bloquer Express »	true, false	false	Permet la création de volumes io2 Block Express.
"kmsKeyId"			L'ARN complet de la clé à utiliser lors du chiffrement du volume. Si elle n'est pas spécifiée, AWS utilisera la clé KMS par défaut pour la région dans laquelle se trouve le volume. Il s'agira d'une clé générée automatiq uement appelée `/aws/ebs` si elle n'est pas modifiée.

Paramètres	Valeurs	Par défaut	Description
« Taille du bloc »			Taille de bloc à utiliser lors du formatage du système de fichiers sous-jacent. Uniquement pris en charge sur les nœuds Linux et avec fstype`ext2`, `ext3``ext4`, ou. `xfs`
« Taille du nœud »			Taille d'inode à utiliser lors du formatage du système de fichiers sous-jacent. Uniquement pris en charge sur les nœuds Linux et avec fstype`ext2`, `ext3``ext4`, ou. `xfs`
"bytesPerInode"			`bytes-per-inode`À utiliser lors du formatage du système de fichiers sous-jacent. Supporté uniquement sur les nœuds Linux et avec fstype`ext2`,,`ext3`. `ext4`

Paramètres	Valeurs	Par défaut	Description
"numberOfInodes"			`number-of-inodes`À utiliser lors du formatage du système de fichiers sous-jacent. Supporté uniquement sur les nœuds Linux et avec fstype`ext2`,,`ext3`. `ext4`
« poste 4 BigAlloc »	true, false	false	Modifie le système de `ext4` fichiers pour qu'il utilise l'allocation de blocs en cluster en activant l'option de `bigalloc` formatage . Avertissement : `bigalloc` il est possible que le noyau Linux de votre nœud ne soit pas totalement compatible.
« poste 4 ClusterSize »			Taille de cluster à utiliser lors du formatage d'un `ext4` système de fichiers lorsque la `bigalloc` fonctionnalité est activée. Remarque : Le `ext4BigAlloc` paramètre doit être défini sur true.

Pour plus d'informations, voir le pilote AWS EBS CSI activé. GitHub

Considérations

> **Note**
>
> Vous pouvez uniquement déployer des charges de travail en fonction du mode automatique EKS StorageClasses sur les nœuds du mode automatique EKS. Si vous disposez d'un cluster comportant plusieurs types de nœuds, vous devez configurer vos charges de travail pour qu'elles s'exécutent uniquement sur les nœuds du mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Déploiement du contrôle".

La capacité de stockage par blocs du mode automatique EKS est différente de celle du pilote EBS CSI.

- Provisionnement statique
  - Si vous souhaitez utiliser des volumes EBS créés en externe avec le mode automatique EKS, vous devez ajouter manuellement une AWS balise avec la clé `eks:eks-cluster-name` et la valeur du nom du cluster.
- Teinte de démarrage du nœud
  - Vous ne pouvez pas utiliser la fonction d'altération du démarrage du nœud pour empêcher la planification des pods avant que la capacité de stockage ne soit prête
- Balises personnalisées sur les volumes approvisionnés dynamiquement
  - Vous ne pouvez pas utiliser l'indicateur de CLI extra-tag pour configurer des balises personnalisées sur des volumes EBS approvisionnés dynamiquement
  - Vous pouvez utiliser le `StorageClass` balisage pour ajouter des balises personnalisées. Le mode automatique EKS ajoutera des balises aux AWS ressources associées. Vous devrez mettre à jour le rôle IAM du cluster pour les balises personnalisées. Pour de plus amples informations, veuillez consulter the section called "AWS Balises personnalisées pour les ressources EKS Auto".
- Mesures de performance détaillées d'EBS
  - Vous ne pouvez pas accéder aux métriques Prometheus pour les performances détaillées d'EBS

Installation du module complémentaire CSI Snapshot Controller

Le mode automatique EKS est compatible avec le module complémentaire Amazon EKS du CSI Snapshot Controller.

AWS vous suggère de configurer ce module complémentaire pour qu'il s'exécute sur le pool de `system` nœuds intégré.

Pour plus d'informations, consultez :

- the section called "Exécutez des modules complémentaires essentiels"
- the section called "Passez en revue les pools de nœuds intégrés"
- the section called "Contrôleur d'instantané CSI"

Pour installer le contrôleur Snapshot dans le pool de nœuds du système

1. Ouvrez votre cluster EKS dans la AWS console
2. Dans l'onglet Modules complémentaires, sélectionnez Obtenir plus de modules complémentaires
3. Sélectionnez le CSI Snapshot Controller, puis Next
4. Sur la page Configurer les paramètres des modules complémentaires sélectionnés, sélectionnez Paramètres de configuration facultatifs pour afficher le schéma de configuration des modules complémentaires
   a. Insérez le yaml suivant pour associer le contrôleur de capture instantanée au pool de `system` nœuds. Le contrôleur de capture d'écran inclut une tolérance pour les `CriticalAddonsOnly` taches.

   ```
   {
           "nodeSelector": {
               "karpenter.sh/nodepool": "system"
           }
   }
   ```

   b. Sélectionnez Next
5. Vérifiez la configuration du module complémentaire, puis sélectionnez Créer

Désactiver le mode automatique EKS

Vous pouvez désactiver le mode automatique EKS sur un cluster EKS existant. Il s'agit d'une opération destructrice.

- EKS mettra fin à toutes les EC2 instances gérées par le mode automatique EKS.
- EKS supprimera tous les équilibreurs de charge fonctionnant en mode automatique EKS.
- EKS ne supprimera pas les volumes EBS approvisionnés par le mode automatique d'EKS.

Le mode automatique EKS est conçu pour gérer entièrement les ressources qu'il crée. Des interventions manuelles peuvent empêcher le mode automatique d'EKS de nettoyer complètement ces ressources lorsqu'il est désactivé. Par exemple, si vous avez fait référence à un groupe de sécurité géré à partir de règles de groupe de sécurité externes et que vous oubliez de supprimer cette référence avant de désactiver le mode automatique EKS pour un cluster, le groupe de sécurité géré sera divulgué (il ne sera pas supprimé). Les étapes ci-dessous décrivent comment supprimer un groupe de sécurité divulgué si cela devait se produire.

Désactiver le mode automatique EKS (AWS console)

1. Ouvrez la page de présentation de votre cluster dans le AWS Management Console.
2. Sous le mode automatique EKS, sélectionnez Gérer
3. Basculez le mode EKS Auto sur`off`.

Si aucun groupe de sécurité géré n'est supprimé à la fin de ce processus, vous pouvez le supprimer manuellement à l'aide des descriptions de la section Supprimer un groupe de sécurité.

Désactiver le mode automatique EKS (AWS CLI)

Utilisez la commande suivante pour désactiver le mode automatique EKS sur un cluster existant.

La `aws` CLI doit être installée et vous devez être connecté avec des autorisations suffisantes pour gérer les clusters EKS. Pour de plus amples informations, veuillez consulter Configuration.

> **Note**
>
> Les fonctionnalités de calcul, de stockage par blocs et d'équilibrage de charge doivent toutes être activées ou désactivées dans la même demande.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=false \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": false}}' \
 --storage-config '{"blockStorage":{"enabled": false}}'
```

Vous pouvez vérifier si un groupe de sécurité en mode automatique EKS divulgué n'a pas pu être supprimé après avoir désactivé le mode automatique EKS comme suit :

```
aws ec2 describe-security-groups \
    --filters Name=tag:eks:eks-cluster-name,Values=<cluster-Name> Name=tag-
key,Values=ingress.eks.amazonaws.com/resource,service.eks.amazonaws.com/resource --
query "SecurityGroups[*].[GroupName]"
```

Pour supprimer ensuite le groupe de sécurité :

```
aws ec2 delete-security-group --group-name=<sg-name>
```

Mettre à jour la version Kubernetes d'un cluster en mode automatique EKS

Cette rubrique explique comment mettre à jour la version Kubernetes de votre cluster en mode automatique. Le mode automatique simplifie le processus de mise à jour des versions en coordonnant les mises à jour du plan de contrôle avec les remplacements de nœuds, tout en maintenant la disponibilité de la charge de travail grâce aux budgets d'interruption des modules.

Lors de la mise à niveau d'un cluster en mode automatique, de nombreux composants qui nécessitaient traditionnellement des mises à jour manuelles sont désormais gérés dans le cadre du service. La compréhension des aspects automatisés du processus de mise à niveau et de vos responsabilités permet de garantir une transition de version fluide pour votre cluster.

En savoir plus sur les mises à jour avec le mode automatique EKS

Une fois que vous avez lancé une mise à niveau du plan de contrôle, le mode automatique EKS met à niveau les nœuds de votre cluster. Lorsque les nœuds expirent, le mode automatique EKS les remplace par de nouveaux nœuds. Les nouveaux nœuds possèdent la nouvelle version de Kubernetes correspondante. Le mode automatique EKS respecte les budgets d'interruption des pods lors de la mise à niveau des nœuds.

De plus, vous n'avez plus besoin de mettre à jour des composants tels que :

- CNI Amazon VPC
- AWS Contrôleur Load Balancer
- CoreDNS
- `kube-proxy`
- Karpenter
- AWS pilote EBS CSI

Le mode automatique EKS remplace ces composants par des fonctionnalités de service.

Vous êtes toujours responsable de la mise à jour :

- Applications et charges de travail déployées sur votre cluster
- Modules complémentaires et contrôleurs autogérés
- Modules complémentaires Amazon EKS
  - Apprenez comment the section called "Mettre à jour un module complémentaire"

Découvrez les meilleures pratiques pour les mises à niveau de clusters

Démarrer la mise à jour du cluster

Pour démarrer une mise à jour du cluster, voirthe section called "Mettre à jour la version de Kubernetes".

Activer ou désactiver la fonction intégrée NodePools

Le mode automatique EKS en possède deux intégrés NodePools. Vous pouvez les activer ou les désactiver à NodePools l'aide de la AWS console, de la CLI ou de l'API.

NodePool Référence intégrée

- `system`
  - Cela NodePool a une `CriticalAddonsOnly` odeur. De nombreux modules complémentaires EKS, tels que CoreDNS, tolèrent cette altération. Utilisez ce pool de nœuds du système pour séparer les applications critiques du cluster.
  - Supporte à la fois les `arm64` architectures `amd64` et les architectures.
- `general-purpose`

- Cela NodePool permet de lancer des nœuds pour des charges de travail générales dans votre cluster.
- Utilise uniquement `amd64` l'architecture.

Les deux sont intégrés NodePools :

- Utiliser l'EKS par défaut NodeClass
- Utiliser uniquement la EC2 capacité à la demande
- Utiliser les familles d' EC2 instances C, M et R
- Exiger des EC2 instances de génération 5 ou plus récentes

> **Note**
>
> L'activation d'au moins une fonction intégrée NodePool est requise pour qu'EKS puisse configurer la « valeur par défaut » NodeClass. Si vous désactivez toutes les NodePools fonctionnalités intégrées, vous devrez créer une personnalisation NodeClass et en configurer une NodePool pour l'utiliser. Pour plus d'informations sur NodeClasses, voirthe section called "Créer une classe de nœuds".

Procédure

Prérequis

- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
  - Connectez-vous à la CLI avec des autorisations IAM suffisantes pour créer des AWS ressources, notamment des politiques IAM, des rôles IAM et des clusters EKS.

Activer avec la AWS CLI

Utilisez la commande suivante pour activer les deux fonctions intégrées NodePools :

```
aws eks update-cluster-config \
```

```
  --name <cluster-name> \
  --compute-config '{
    "nodeRoleArn": "<node-role-arn>",
    "nodePools": ["general-purpose", "system"],
    "enabled": true
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}
  }' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

Vous pouvez modifier la commande pour activer de manière sélective le NodePools.

Désactiver avec la AWS CLI

Utilisez la commande suivante pour désactiver les deux fonctions intégrées NodePools :

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
  "enabled": true,
  "nodePools": []
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

Contrôler si une charge de travail est déployée sur les nœuds du mode automatique EKS

Lorsque vous exécutez des charges de travail dans un cluster EKS avec le mode automatique EKS, vous devrez peut-être contrôler si des charges de travail spécifiques s'exécutent sur les nœuds du mode automatique EKS ou sur d'autres types de calcul. Cette rubrique explique comment utiliser les sélecteurs de nœuds et les règles d'affinité pour garantir que vos charges de travail sont planifiées sur l'infrastructure informatique prévue.

Les exemples présentés dans cette rubrique montrent comment utiliser l'`eks.amazonaws.com/compute-type`étiquette pour exiger ou empêcher le déploiement de la charge de travail sur les

nœuds du mode automatique EKS. Cela est particulièrement utile dans les clusters en mode mixte dans lesquels vous exécutez à la fois le mode automatique EKS et d'autres types de calcul, tels que les approvisionneurs Karpenter autogérés ou les groupes de nœuds gérés par EKS.

Les nœuds du mode automatique EKS ont défini la valeur de l'étiquette sur `eks.amazonaws.com/compute-typeauto`. Vous pouvez utiliser cette étiquette pour contrôler si une charge de travail est déployée sur des nœuds gérés par le mode automatique EKS.

Exiger qu'une charge de travail soit déployée sur les nœuds du mode automatique EKS

> **Note**
>
> Cette `nodeSelector` valeur n'est pas requise pour le mode automatique EKS. Cette `nodeSelector` valeur n'est pertinente que si vous exécutez un cluster en mode mixte, les types de nœuds ne sont pas gérés par le mode automatique EKS. Par exemple, vous pouvez avoir une capacité de calcul statique déployée sur votre cluster avec des groupes de nœuds gérés par EKS, et une capacité de calcul dynamique gérée par le mode automatique EKS.

Vous pouvez l'ajouter `nodeSelector` aux déploiements ou à d'autres charges de travail pour obliger Kubernetes à les planifier sur des nœuds EKS Auto Mode.

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    nodeSelector:
      eks.amazonaws.com/compute-type: auto
```

Exiger qu'une charge de travail ne soit pas déployée sur les nœuds du mode automatique EKS

Vous pouvez l'ajouter `nodeAffinity` aux déploiements ou à d'autres charges de travail pour empêcher Kubernetes de les planifier sur les nœuds du mode automatique EKS.

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
```

```
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - auto
```

Exécutez des modules complémentaires critiques sur des instances dédiées

Dans cette rubrique, vous allez apprendre à déployer une charge de travail avec une `CriticalAddonsOnly` tolérance afin que le mode automatique EKS la planifie sur le pool de `system` nœuds.

Le pool de `system` nœuds intégré à EKS Auto Mode est conçu pour exécuter des modules complémentaires essentiels sur des instances dédiées. Cette séparation garantit que les composants essentiels disposent de ressources dédiées et sont isolés des charges de travail générales, améliorant ainsi la stabilité et les performances globales du cluster.

Ce guide explique comment déployer des modules complémentaires dans le pool de `system` nœuds en utilisant la `CriticalAddonsOnly` tolérance et les sélecteurs de nœuds appropriés. En suivant ces étapes, vous pouvez vous assurer que vos applications critiques sont planifiées sur les `system` nœuds dédiés, en tirant parti des avantages d'isolation et d'allocation des ressources fournis par la structure de pool de nœuds spécialisée d'EKS Auto Mode.

Le mode automatique EKS possède deux pools de nœuds intégrés : `general-purpose` et`system`. Pour de plus amples informations, veuillez consulter the section called "Passez en revue les pools de nœuds intégrés".

L'objectif du pool de `system` nœuds est de séparer les modules complémentaires critiques sur différents nœuds. Les nœuds provisionnés par le pool de `system` nœuds présentent une empreinte `CriticalAddonsOnly` Kubernetes. Kubernetes ne planifiera des pods sur ces nœuds que s'ils disposent d'une tolérance correspondante. Pour plus d'informations, consultez Taints and Tolerations dans la documentation de Kubernetes.

Prérequis

- Cluster en mode automatique EKS avec le pool de `system` nœuds intégré activé. Pour plus d'informations, consultez the section called "Passez en revue les pools de nœuds intégrés".

- `kubectl`installé et configuré. Pour de plus amples informations, veuillez consulter Configuration.

Procédure

Consultez l'exemple de fichier yaml ci-dessous. Notez les configurations suivantes :

- `nodeSelector`— Cela associe la charge de travail au pool de `system` nœuds intégré. Ce pool de nœuds doit être activé avec l' AWS API. Pour de plus amples informations, veuillez consulter the section called "Passez en revue les pools de nœuds intégrés".
- `tolerations`— Cette tolérance permet de remédier à l'altération des `CriticalAddonsOnly` nœuds du pool de `system` nœuds.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      nodeSelector:
        karpenter.sh/nodepool: system
      tolerations:
      - key: "CriticalAddonsOnly"
        operator: "Exists"
      containers:
      - name: app
        image: nginx:latest
        resources:
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Pour mettre à jour une charge de travail à exécuter sur le pool de `system` nœuds, vous devez :

1. Mettez à jour la charge de travail existante pour ajouter les configurations suivantes décrites ci-dessus :
   - `nodeSelector`
   - `tolerations`
2. Déployez la charge de travail mise à jour sur votre cluster avec `kubectl apply`

Après avoir mis à jour la charge de travail, elle s'exécutera sur des nœuds dédiés.

Utiliser les politiques réseau avec le mode automatique EKS

Les politiques réseau vous permettent de contrôler le flux de trafic au niveau de l'adresse IP ou du port au sein de votre cluster Amazon EKS. Cette rubrique explique comment activer et utiliser les politiques réseau avec le mode automatique d'EKS.

Prérequis

- Un cluster Amazon EKS avec le mode automatique EKS activé
- kubectl configuré pour se connecter à votre cluster

Étape 1 : activer le Network Policy Controller

Pour utiliser les politiques réseau avec le mode automatique EKS, vous devez d'abord activer le Network Policy Controller en appliquant un ConfigMap à votre cluster.

1. Créez un fichier nommé `enable-network-policy.yaml` avec le contenu suivant:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

2. Appliquez le ConfigMap à votre cluster :

```
kubectl apply -f enable-network-policy.yaml
```

Étape 2 : activer les politiques réseau dans la classe de nœuds

Avant de pouvoir utiliser les politiques réseau, vous devez vous assurer que votre classe de nœud est configurée pour les prendre en charge. Procédez comme suit :

1. Créez ou modifiez un fichier YAML de classe de nœud (par exemple`nodeclass-network-policy.yaml`) avec le contenu suivant :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: network-policy-enabled
spec:
  # Enables network policy support
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. Appliquez la configuration de classe de nœuds à votre cluster :

```
kubectl apply -f nodeclass-network-policy.yaml
```

3. Vérifiez que la classe Node a été créée :

```
kubectl get nodeclass network-policy-enabled
```

4. Mettez à jour votre pool de nœuds pour utiliser cette classe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Créer un pool de nœuds".

Une fois que vos nœuds utiliseront cette classe de nœuds, ils seront en mesure d'appliquer les politiques réseau. Vous pouvez maintenant créer et appliquer des politiques réseau pour contrôler le trafic au sein de votre cluster. Pour toutes les options de configuration des classes de nœuds, consultezthe section called "Créer une classe de nœuds".

Étape 3 : créer et tester des politiques réseau

Votre cluster EKS Auto Mode est désormais configuré pour prendre en charge les politiques réseau Kubernetes. Vous pouvez le tester avec lethe section called "Démonstration de politique Stars".

Sous-réseaux de balises pour le mode automatique EKS

Si vous utilisez la fonctionnalité d'équilibrage de charge du mode automatique d'EKS, vous devez ajouter des AWS balises à vos sous-réseaux VPC.

Contexte

Ces balises identifient les sous-réseaux associés au cluster, et surtout si le sous-réseau est public ou privé.

Les sous-réseaux publics ont un accès direct à Internet via une passerelle Internet. Ils sont utilisés pour les ressources qui doivent être accessibles au public, telles que les équilibreurs de charge.

Les sous-réseaux privés n'ont pas d'accès direct à Internet et utilisent des passerelles NAT pour le trafic sortant. Ils sont utilisés pour les ressources internes telles que les nœuds EKS qui n'ont pas besoin d'être publics IPs.

Pour en savoir plus sur les passerelles NAT et les passerelles Internet, consultez Connect your VPC to other networks dans le manuel Amazon Virtual Private Cloud (VPC) User Guide.

Exigence

À l'heure actuelle, les sous-réseaux utilisés pour l'équilibrage de charge par le mode automatique d'EKS doivent comporter l'une des balises suivantes.

Sous-réseaux publics

Les sous-réseaux publics sont utilisés pour les équilibreurs de charge connectés à Internet. Ces sous-réseaux doivent comporter les balises suivantes :

Clé	Valeur
`kubernetes.io/role/elb`	`1`ou ``

Sous-réseaux privés

Les sous-réseaux privés sont utilisés pour les équilibreurs de charge internes. Ces sous-réseaux doivent comporter les balises suivantes :

Clé	Valeur
`kubernetes.io/role/internal-elb`	1ou ``

Procédure

Avant de commencer, identifiez les sous-réseaux publics (avec accès Internet Gateway) et ceux qui sont privés (à l'aide de NAT Gateway). Vous aurez besoin d'autorisations pour modifier les ressources VPC.

AWS Management Console

1. Ouvrez la console Amazon VPC et accédez à Subnets
2. Sélectionnez le sous-réseau à étiqueter
3. Choisissez l'onglet Tags et sélectionnez Ajouter un tag
4. Ajoutez la balise appropriée :
   - Pour les sous-réseaux publics : Key= `kubernetes.io/role/elb`
   - Pour les sous-réseaux privés : Key= `kubernetes.io/role/internal-elb`
5. Définir la valeur sur 1 ou laisser le champ vide
6. Enregistrez et répétez pour les sous-réseaux restants

AWS CLI

Pour les sous-réseaux publics :

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/elb,Value=1
```

Pour les sous-réseaux privés :

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/internal-elb,Value=1
```

`subnet-ID`Remplacez-le par votre identifiant de sous-réseau actuel.

Déployez une charge de travail accélérée

Ce didacticiel montre comment le mode automatique d'Amazon EKS simplifie le lancement de charges de travail accélérées. Le mode automatique d'Amazon EKS rationalise les opérations au-delà du cluster lui-même en automatisant les principaux composants de l'infrastructure en fournissant des fonctionnalités de calcul, de mise en réseau, d'équilibrage de charge, de stockage ainsi que d'accès et de gestion des identités prêtes à l'emploi.

Le mode automatique Amazon EKS inclut les pilotes et les plug-ins de périphérique requis pour certains types d'instances, tels que les pilotes NVIDIA et AWS Neuron. Il n'est pas nécessaire d'installer ou de mettre à jour ces composants.

Le mode automatique EKS gère automatiquement les pilotes des accélérateurs suivants :

- AWS Trainium
- AWS Inférentie
- Instances EC2 accélérées NVIDIA GPUs sur Amazon

> **Note**
>
> Le mode automatique EKS inclut le plug-in d'appareil NVIDIA pour Kubernetes. Ce plugin s'exécute automatiquement et n'est pas visible en tant que daemon défini dans votre cluster.

Support réseau supplémentaire :

- Adaptateur Elastic Fabric (EFA)

Le mode automatique d'Amazon EKS élimine les difficultés liées à la gestion du pilote d'accélérateur et des plug-ins de périphérique.

Vous pouvez également réaliser des économies en réduisant le cluster à zéro. Vous pouvez configurer le mode automatique d'EKS pour mettre fin aux instances lorsqu'aucune charge de travail n'est en cours d'exécution. Cela est utile pour les charges de travail d'inférence basées sur des lots.

Vous trouverez ci-dessous un exemple de lancement de charges de travail accélérées avec le mode automatique d'Amazon EKS.

Prérequis

- Un cluster Kubernetes avec le mode automatique Amazon EKS configuré.
- Une classe de nœuds `default` EKS telle que créée lorsque le `general-purpose` ou les pools de nœuds `system` gérés sont activés.

Étape 1 : Déployer une charge de travail GPU

Dans cet exemple, vous allez créer un NodePool pour les charges de travail basées sur NVIDIA qui nécessitent 45 Go de mémoire GPU. Avec le mode automatique d'EKS, vous utilisez les contraintes de planification de Kubernetes pour définir les exigences de votre instance.

Pour déployer le mode automatique Amazon EKS `NodePool` et l'exemple`workload`, consultez les informations suivantes NodePool et la définition du pod, puis enregistrez-les sous `nodepool-gpu.yaml` et `pod.yaml` :

nodepool-gpu.yaml

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
```

```
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
```

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    eks.amazonaws.com/instance-gpu-name: l40s
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Notez que le `eks.amazonaws.com/compute-type: auto` sélecteur nécessite que la charge de travail soit déployée sur un nœud Amazon EKS Auto Mode. Cela crée NodePool également une odeur qui permet uniquement de planifier des pods avec des tolérances GPUs pour Nvidia.

Appliquez la charge de travail NodePool et à votre cluster.

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

Vous devriez voir la sortie suivante :

```
nodepool.karpenter.sh/gpu configured created
pod/nvidia-smi created
```

Patientez quelques secondes et vérifiez les nœuds de votre cluster. Vous devriez maintenant voir un nouveau nœud provisionné dans votre cluster Amazon EKS Auto Mode :

```
> kubectl get nodes

NAME        TYPE          CAPACITY    ZONE         NODE                   READY   AGE
gpu-dnknr   g6e.2xlarge   on-demand   us-west-2b   i-02315c7d7643cdee6   True    76s
```

Étape 2 : Valider

Vous pouvez constater qu'Amazon EKS Auto Mode a été lancé au `g6e.2xlarge` lieu d'un `g6.2xlarge` car la charge de travail nécessitait une instance avec des l40sGPU, conformément aux contraintes de planification Kubernetes suivantes :

```
...
  nodeSelector:
    eks.amazonaws.com/instance-gpu-name: l40s
...
    requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
```

Maintenant, examinez les journaux des conteneurs en exécutant la commande suivante :

```
kubectl logs nvidia-smi
```

Exemple de sortie :

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 535.230.02             Driver Version: 535.230.02   CUDA Version: 12.2
 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA L40S                    On  | 00000000:30:00.0 Off |                    0
 |
| N/A   27C    P8              23W / 350W |      0MiB / 46068MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------+----------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
|  GPU   GI   CI        PID   Type   Process name                          GPU Memory
 |
|        ID   ID                                                           Usage
 |
|
=========================================================================================|
|  No running processes found
 |
+-----------------------------------------------------------------------------------------
+
```

Vous pouvez constater que le conteneur a détecté qu'il s'exécute sur une instance dotée d'un `NVIDIA` processeur graphique et que vous n'avez pas eu à installer de pilote de périphérique, car cela est géré par le mode automatique Amazon EKS.

Étape 3 : Nettoyage

Pour supprimer tous les objets créés, utilisez `kubectl` pour supprimer l'exemple de déploiement NodePool afin de mettre fin au nœud :

```
kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml
```

Exemple de NodePools référence

Créez une carte NVIDIA NodePool

Les définitions suivantes sont NodePool les suivantes :

- Lancez uniquement des instances de `g6e` et de `g6` la famille
- Consolidez les nœuds lorsqu'ils sont vides pendant 1 heure
  - La valeur d'une heure pour `consolodateAfter` prend en charge les charges de travail élevées et réduit le taux de désabonnement des nœuds. Vous pouvez effectuer le réglage `consolidateAfter` en fonction de vos exigences en matière de charge de travail.

Exemple NodePool de famille d'instances GPU et de consolidation

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
```

```
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      terminationGracePeriod: 24h0m0s
```

Au lieu de définir le, `eks.amazonaws.com/instance-gpu-name` vous pouvez l'utiliser `eks.amazonaws.com/instance-family` pour spécifier la famille d'instances. Pour d'autres labels connus qui influencent la révision de la planification, voirthe section called "Étiquettes compatibles avec le mode automatique EKS".

Si vous avez des besoins de stockage spécifiques, vous pouvez ajuster le stockage `iops` éphémère des nœuds `size` et `throughput` en créant le vôtre NodeClassà référencer dans le. NodePool En savoir plus sur les NodeClass options configurables.

Exemple de configuration de stockage pour NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

Définition d'un AWS trainium et AWS d'une inférence NodePool

Ce qui suit NodePool contient un `eks.amazonaws.com/instance-category` ensemble qui indique de ne lancer que les instances de la famille Inferentia et Trainium :

```
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values:
```

```
            - inf
            - trn
```

Générez des rapports de conformité CIS à partir des nœuds Kubernetes à l'aide de kubectl debug

Cette rubrique décrit comment générer des rapports de conformité CIS (Center for Internet Security) pour les nœuds Amazon EKS à l'aide de la `kubectl debug` commande. La commande vous permet de créer temporairement un conteneur de débogage sur un nœud Kubernetes et d'exécuter des contrôles de conformité CIS à l'aide de l'outil. `apiclient` L'`apiclient`outil fait partie de Bottlerocket OS, le système d'exploitation utilisé par les nœuds EKS Auto Mode.

Prérequis

Avant de commencer, assurez-vous d'avoir :

- Accès à un cluster Amazon EKS `kubectl` configuré (la version doit être au moins v1.32.0 ; type `kubectl version` pour vérifier).
- Les autorisations IAM appropriées pour déboguer les nœuds.
- Un profil valide qui autorise les opérations de débogage (par exemple,`sysadmin`).

Pour plus d'informations sur l'utilisation de profils de débogage avec`kubectl`, consultez la section Débogage d'un pod ou d'un nœud lors de l'application d'un profil dans la documentation de Kubernetes.

Procédure

1. Déterminez l'ID d' AWS instance du nœud sur lequel vous souhaitez exécuter le rapport. Utilisez la commande suivante pour répertorier les nœuds du cluster. L'ID de l'instance se trouve dans la colonne du nom et commence par `i-` :

   ```
   kubectl get nodes
   ```

   ```
   NAME                  STATUS   ROLES    AGE   VERSION
   i-0ea0ba0f8ef9ad609   Ready    <none>   62s   v1.30.10-eks-1a9dacd
   ```

2. Exécutez la commande suivante, en `<instance-id>` remplaçant par l'ID d'instance du nœud que vous souhaitez interroger :

```
kubectl debug node/<instance-id> -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023 -- bash -c "yum install -q -y util-linux-core; nsenter -
t 1 -m apiclient report cis --level 1 --format text"
```

Les composants de cette commande sont les suivants :

- `kubectl debug node/<instance-id>`— Crée une session de débogage sur l'ID d' EC2 instance spécifié.
- `-it`— Alloue un TTY (shell de ligne de commande) et maintient stdin ouvert pour une utilisation interactive.
- `--profile=sysadmin`— Utilise le `kubectl` profil spécifié avec les autorisations appropriées.
- `--image=public.ecr.aws/amazonlinux/amazonlinux:2023`— À utiliser `amazonlinux:2023` comme image de conteneur pour le débogage.
- `bash -c "…"` — Exécute les commandes suivantes dans un shell bash :
  - `yum install -q -y util-linux-core`— Installe discrètement le package d'utilitaires requis.
  - `nsenter -t 1 -m`— S'exécute `nsenter` pour entrer dans l'espace de noms du processus hôte (PID 1).
  - `apiclient report cis --level 1 --format text`— Exécute le rapport de conformité CIS au niveau 1 avec sortie texte.

3. Passez en revue le texte de sortie du rapport.

Interprétation de la sortie

La commande génère un rapport sous forme de texte indiquant l'état de conformité des différents contrôles CIS. La sortie inclut :

- Contrôle CIS individuel IDs
- Description de chaque commande
- État de réussite, d'échec ou d'échec pour chaque vérification
- Détails expliquant les éventuels problèmes de conformité

Voici un exemple de sortie du rapport exécuté sur une instance de Bottlerocket :

```
Benchmark name:  CIS Bottlerocket Benchmark
```

```
Version:         v1.0.0
Reference:       https://www.cisecurity.org/benchmark/bottlerocket
Benchmark level: 1
Start time:      2025-04-11T01:40:39.055623436Z

[SKIP] 1.2.1     Ensure software update repositories are configured (Manual)
[PASS] 1.3.1     Ensure dm-verity is configured (Automatic)[PASS] 1.4.1     Ensure
 setuid programs do not create core dumps (Automatic)
[PASS] 1.4.2     Ensure address space layout randomization (ASLR) is enabled
 (Automatic)
[PASS] 1.4.3     Ensure unprivileged eBPF is disabled (Automatic)
[PASS] 1.5.1     Ensure SELinux is configured (Automatic)
[SKIP] 1.6       Ensure updates, patches, and additional security software are
 installed (Manual)
[PASS] 2.1.1.1   Ensure chrony is configured (Automatic)
[PASS] 3.2.5     Ensure broadcast ICMP requests are ignored (Automatic)
[PASS] 3.2.6     Ensure bogus ICMP responses are ignored (Automatic)
[PASS] 3.2.7     Ensure TCP SYN Cookies is enabled (Automatic)
[SKIP] 3.4.1.3   Ensure IPv4 outbound and established connections are configured
 (Manual)
[SKIP] 3.4.2.3   Ensure IPv6 outbound and established connections are configured
 (Manual)
[PASS] 4.1.1.1   Ensure journald is configured to write logs to persistent disk
 (Automatic)
[PASS] 4.1.2     Ensure permissions on journal files are configured (Automatic)

Passed:          11
Failed:          0
Skipped:         4
Total checks:    15
```

Pour plus d'informations sur le benchmark, consultez Kubernetes Benchmark du Center for Internet Security (CIS).

Ressources connexes

- Référence Bottlerocket CIS dans la documentation du système d'exploitation Bottlerocket.
- Déboguez les pods en cours d'exécution dans la documentation de Kubernetes.
- Benchmark Kubernetes établi par le Center for Internet Security (CIS)

Activez le chiffrement des volumes EBS avec des clés KMS gérées par le client pour le mode automatique EKS

Vous pouvez chiffrer le volume racine éphémère des instances du mode automatique EKS à l'aide d'une clé KMS gérée par le client.

Le mode automatique Amazon EKS utilise des rôles liés à des services pour déléguer des autorisations à d'autres AWS services lors de la gestion de volumes EBS chiffrés pour vos clusters Kubernetes. Cette rubrique explique comment configurer la politique de clé dont vous avez besoin lorsque vous spécifiez une clé gérée par le client pour le chiffrement Amazon EBS avec le mode automatique EKS.

Considérations :

- Le mode automatique EKS n'a pas besoin d'autorisation supplémentaire pour utiliser la clé AWS gérée par défaut afin de protéger les volumes chiffrés de votre compte.
- Cette rubrique traite du chiffrement des volumes éphémères, les volumes racines pour les instances. EC2 Pour plus d'informations sur le chiffrement des volumes de données utilisés pour les charges de travail, consultez. the section called "Créez StorageClass"

Présentation

Les clés AWS KMS suivantes peuvent être utilisées pour le chiffrement du volume racine Amazon EBS lorsque le mode automatique EKS lance des instances :

- AWS clé gérée : clé de chiffrement de votre compte créée, détenue et gérée par Amazon EBS. Il s'agit de la clé de chiffrement par défaut d'un nouveau compte.
- Clé gérée par le client : clé de chiffrement personnalisée que vous créez, détenez et gérez.

> **Note**
>
> La clé doit être symétrique. Amazon EBS ne prend pas en charge les clés asymétriques gérées par le client.

Étape 1 : Configuration de la politique clé

Vos clés KMS doivent avoir une politique clé qui autorise le mode automatique EKS à lancer des instances avec des volumes Amazon EBS chiffrés à l'aide d'une clé gérée par le client.

Configurez votre politique clé avec la structure suivante :

> **Note**
>
> Cette politique inclut uniquement les autorisations pour le mode automatique EKS. La politique clé peut nécessiter des autorisations supplémentaires si d'autres identités doivent utiliser la clé ou gérer les subventions.

```
{
    "Version": "2012-10-17",
    "Id": "MyKeyPolicy",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws: iam::<account-id>:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws: iam::<account-id>:role/ClusterServiceRole"
                ]
```

```
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Assurez-vous de le remplacer `<account-id>` par votre numéro de AWS compte actuel.

Lors de la configuration de la politique clé :

- Ils `ClusterServiceRole` doivent disposer des autorisations IAM nécessaires pour utiliser la clé KMS pour les opérations de chiffrement.
- Cette `kms:GrantIsForAWSResource` condition garantit que les subventions ne peuvent être créées que pour des AWS services.

Étape 2 : Configuration NodeClass avec votre clé gérée par le client

Après avoir configuré la politique des clés, référencez la clé KMS dans votre NodeClass configuration du mode automatique EKS :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Insert existing configuration

  ephemeralStorage:
    size: "80Gi"  # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000    # Range: 3000-16000
    throughput: 125  # Range: 125-1000
```

```
    # KMS key for encryption
    kmsKeyID: "arn:aws: kms:<region>:<account-id>:key/<key-id>"
```

Remplacez les valeurs de l'espace réservé par vos valeurs réelles :

- `<region>`avec votre AWS région
- `<account-id>`avec votre identifiant AWS de compte
- `<key-id>`avec votre identifiant de clé KMS

Vous pouvez spécifier la clé KMS en utilisant l'un des formats suivants :

- ID de clé KMS : `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- ARN de la clé KMS : `arn:aws: kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- Nom de l'alias clé : `alias/eks-auto-mode-key`
- Alias (ARN) clé : `arn:aws: kms:us-west-2:111122223333:alias/eks-auto-mode-key`

Appliquez la NodeClass configuration à l'aide de kubectl :

```
kubectl apply -f nodeclass.yaml
```

Ressources connexes

- Création d'une classe de nœuds pour Amazon EKS
- Pour plus d'informations, consultez le guide du développeur du service de gestion des AWS clés
  - Autorisations pour les AWS services dans les politiques clés
  - Modifier une politique clé
  - Subventions dans AWS KMS

Mettre à jour les contrôles d'organisation pour le mode automatique EKS

Certains contrôles organisationnels peuvent empêcher le mode automatique d'EKS de fonctionner correctement. Dans ce cas, vous devez mettre à jour ces contrôles pour permettre au mode automatique d'EKS de disposer des autorisations nécessaires pour gérer les EC2 instances en votre nom.

Le mode automatique EKS utilise un rôle de service pour lancer les EC2 instances qui soutiennent les nœuds du mode automatique EKS. Un rôle de service est un rôle IAM créé dans votre compte et censé être exécuté par un service en votre nom. Les politiques de contrôle des services (SCPs) s'appliquent toujours aux actions effectuées avec des rôles de service. Cela permet à un SCP d'inhiber les opérations du mode automatique. L'occurrence la plus courante se produit lorsqu'un SCP est utilisé pour restreindre les Amazon Machine Images (AMIs) qui peuvent être lancées. Pour permettre au mode automatique EKS de fonctionner, modifiez le SCP afin de permettre le lancement à AMIs partir des comptes du mode automatique EKS.

Vous pouvez également utiliser la AMIs fonctionnalité EC2 Autorisé pour limiter la visibilité sur AMIs d'autres comptes. Si vous utilisez cette fonctionnalité, vous devez étendre les critères d'image pour inclure également les comptes AMI du mode automatique EKS dans les régions qui vous intéressent.

Exemple de SCP pour tout bloquer AMIs sauf le mode automatique EKS AMIs

Le SCP ci-dessous empêche les appels `ec2:RunInstances` sauf si l'AMI appartient au compte EKS Auto Mode AMI pour us-west-2 ou us-east-1.

> **Note**
>
> Il est important de ne pas utiliser la clé de `ec2:Owner` contexte. Amazon possède les comptes AMI EKS Auto Mode et la valeur de cette clé sera toujours la même`amazon`. La construction d'un SCP qui permet le `ec2:Owner` lancement AMIs s'il permet de lancer n'importe quelle AMI appartenant à Amazon, et pas seulement celles destinées au mode automatique d'EKS. `amazon` *

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
```

```
            ]
          }
        }
      }
    ]
}
```

Comptes AMI en mode automatique EKS

AWS les comptes qui varient selon les régions hébergent le mode automatique EKS public AMIs.

AWS Région	Compte
af-south-1	471112993317
ap-east-1	590183728416
ap-northeast-1	851725346105
ap-northeast-2	992382805010
ap-northeast-3	891377407544
ap-south-1	975049899075
ap-south-2	590183737426
ap-southeast-1	339712723301
ap-southeast-2	58264376476
ap-southeast-3	471112941769
ap-southeast-4	590183863144
ap-southeast-5	654654202513
ap-sud-est 7	533267217478
ca-central-1	992382439851
ca-west-1	767397959864

eu-central-1	891376953411
eu-central-2	381492036002
eu-north-1	339712696471
eu-south-1	975049955519
eu-south-2	471112620929
eu-west-1	381492008532
eu-west-2	590184142468
eu-west-3	891376969258
il-central-1	590183797093
me-central-1	637423494195
me-south-1	905418070398
mx-central-1	211125506622
sa-east-1	339712709251
us-east-1	992382739861
us-east-2	975050179949
us-west-1	975050035094
us-west-2	767397842682

Adresse IP publique associée

Lorsqu'il `ec2:RunInstances` est appelé, le `AssociatePublicIpAddress` champ de lancement d'une instance est déterminé automatiquement par le type de sous-réseau dans lequel l'instance est lancée. Un SCP peut être utilisé pour faire en sorte que cette valeur soit explicitement définie sur false, quel que soit le type de sous-réseau dans lequel le lancement est lancé. Dans ce cas, le

NodeClass champ `spec.advancedNetworking.associatePublicIPAddress` peut également être défini sur false pour satisfaire aux exigences du SCP.

```
{
      "Sid": "DenyPublicEC2IPAddesses",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
          "BoolIfExists": {
              "ec2:AssociatePublicIpAddress": "true"
          }
      }
  }
```

Contrôlez le déploiement des charges de travail dans les réservations de capacité EC2 à la demande avec le mode automatique EKS

EC2 Les réservations de capacité à la demande (ODCRs) vous permettent de réserver de la capacité de calcul pour vos EC2 instances Amazon dans une zone de disponibilité spécifique, quelle que soit la durée. Lorsque vous utilisez le mode automatique d'EKS, vous souhaiterez peut-être contrôler si vos charges de travail Kubernetes sont déployées sur ces instances réservées afin de maximiser l'utilisation de la capacité préachetée ou de garantir que les charges de travail critiques ont accès à des ressources garanties.

Par défaut, le mode automatique EKS démarre automatiquement en mode ouvert ODCRs. Cependant, en configurant `capacityReservationSelectorTerms` sur un NodeClass, vous pouvez contrôler explicitement les charges de travail utilisées par ODCRs vos charges de travail. Les nœuds approvisionnés à l'aide de la configuration ODCRs auront `karpenter.sh/capacity-type: reserved` et seront priorisés par rapport à la demande et au spot. Une fois cette fonctionnalité activée, le mode automatique d'EKS n'utilisera plus automatiquement les options ouvertes. ODCRs Elles doivent être explicitement sélectionnées par un NodeClass, ce qui vous permet de contrôler précisément l'utilisation des réservations de capacité au sein de votre cluster.

> **Warning**
>
> Si vous effectuez une configuration `capacityReservationSelectorTerms` NodeClass dans un cluster, le mode automatique d'EKS n'utilisera plus automatiquement l'option open ODCRs pour aucun NodeClass élément du cluster.

Exemple NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
spec:
  # Optional: Selects upon on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        app: "my-app"
      # Optional owning account ID filter
      owner: "012345678901"
```

Cet exemple NodeClass illustre deux approches de sélection ODCRs. La première méthode fait directement référence à un ODCR spécifique par son ID (`cr-56fac701cc1951b03`). La seconde méthode utilise la sélection basée sur les balises, le ciblage à l' ODCRs aide de la balise`Name: "targeted-odcr"`. Vous pouvez également éventuellement filtrer en fonction du AWS compte propriétaire de la réservation, ce qui est particulièrement utile dans les scénarios entre comptes ou lorsque vous travaillez avec des réservations à capacité partagée.

Découvrez comment fonctionne le mode automatique EKS

Utilisez ce chapitre pour découvrir comment fonctionnent les composants des clusters en mode automatique Amazon EKS.

Rubriques

- En savoir plus sur les instances gérées en mode automatique Amazon EKS
- En savoir plus sur l'identité et l'accès en mode automatique EKS
- En savoir plus sur la mise en réseau VPC et l'équilibrage de charge en mode automatique EKS

En savoir plus sur les instances gérées en mode automatique Amazon EKS

Cette rubrique explique comment le mode automatique Amazon EKS gère les EC2 instances Amazon dans votre cluster EKS. Lorsque vous activez le mode automatique EKS, les ressources de calcul de votre cluster sont automatiquement provisionnées et gérées par EKS, ce qui modifie la façon dont vous interagissez avec les EC2 instances qui servent de nœuds dans votre cluster.

Il est essentiel de comprendre comment Amazon EKS Auto Mode gère les instances pour planifier votre stratégie de déploiement de charge de travail et vos procédures opérationnelles. Contrairement aux EC2 instances traditionnelles ou aux groupes de nœuds gérés, ces instances suivent un modèle de cycle de vie différent dans lequel EKS assume la responsabilité de nombreux aspects opérationnels, tout en limitant certains types d'accès et de personnalisation.

Le mode automatique Amazon EKS automatise les tâches de routine pour créer de nouvelles EC2 instances et les attache en tant que nœuds à votre cluster EKS. Le mode automatique EKS détecte lorsqu'une charge de travail ne peut pas s'adapter aux nœuds existants et crée une nouvelle EC2 instance.

Le mode automatique Amazon EKS est responsable de la création, de la suppression et de l'application des correctifs aux EC2 instances. Vous êtes responsable des conteneurs et des pods déployés sur l'instance.

EC2 Les instances créées par le mode automatique d'EKS sont différentes des autres EC2 instances, ce sont des instances gérées. Ces instances gérées appartiennent à EKS et sont plus restreintes. Vous ne pouvez pas accéder directement au logiciel ou l'installer sur les instances gérées par le mode automatique EKS.

AWS suggère d'exécuter le mode automatique EKS ou Karpenter autogéré. Vous pouvez installer les deux lors d'une migration ou dans une configuration avancée. Si vous avez installé les deux, configurez vos pools de nœuds de manière à ce que les charges de travail soient associées au mode automatique Karpenter ou EKS.

Pour plus d'informations, consultez les instances EC2 gérées par Amazon dans le guide de EC2 l'utilisateur Amazon.

Tableau comparatif

EC2 Instance standard	Instance gérée en mode automatique EKS
Vous êtes responsable de l'application des correctifs et de la mise à jour de l'instance.	AWS corrige et met à jour automatiquement l'instance.
EKS n'est pas responsable du logiciel installé sur l'instance.	EKS est responsable de certains logiciels présents sur l'instance`kubelet`, tels que le moteur d'exécution du conteneur et le système d'exploitation.
Vous pouvez supprimer l' EC2 instance à l'aide de l' EC2 API.	EKS détermine le nombre d'instances déployées sur votre compte. Si vous supprimez une charge de travail, EKS réduira le nombre d'instances de votre compte.
Vous pouvez utiliser SSH pour accéder à l' EC2 instance.	Vous pouvez déployer des pods et des conteneurs sur l'instance gérée.
Vous déterminez le système d'exploitation et l'image (AMI).	AWS détermine le système d'exploitation et l'image.
Vous pouvez déployer des charges de travail qui reposent sur les fonctionnalités de Windows ou d'Ubuntu.	Vous pouvez déployer des conteneurs basés sur Linux, mais sans dépendances spécifiques au système d'exploitation.
Vous déterminez le type et la famille d'instanc es à lancer.	AWS détermine le type et la famille d'instanc es à lancer. Vous pouvez utiliser un pool de nœuds pour limiter les types d'instances sélectionnés par EKS Auto Mode.

Les fonctionnalités suivantes fonctionnent à la fois pour les instances gérées et les EC2 instances standard :

- Vous pouvez afficher l'instance dans la AWS console.
- Vous pouvez utiliser le stockage d'instance comme stockage éphémère pour les charges de travail.

Support AMI

Avec le mode automatique EKS, AWS détermine l'image (AMI) utilisée pour vos nœuds de calcul. AWS surveille le déploiement des nouvelles versions de l'AMI en mode automatique d'EKS. Si vous rencontrez des problèmes de charge de travail liés à une version d'AMI, créez un dossier de support. Pour plus d'informations, consultez les sections Création de dossiers de support et gestion de dossiers dans le Guide de l'utilisateur du AWS support.

En général, EKS publie une nouvelle AMI chaque semaine contenant des correctifs CVE et de sécurité.

Référence d'instance prise en charge par EKS Auto Mode

Le mode automatique EKS crée uniquement des instances dont les types sont pris en charge et qui répondent à une exigence de taille minimale.

Le mode automatique EKS prend en charge les types d'instances suivants :

Famille	Types d'instances
Optimisé pour le calcul (C)	c8g, c7a, c7g, c7gn, c7gd, c7i, c7i-flex, c6a, c6g, c6i, c6gn, c6id, c6in, c6gd, c5, c5a, c5d, c5ad, c5n, c4
Usage général (M)	m8g, m7i, m7a, m7g, m7gd, m7i-flex, m6a, m6i, m6in, m6idn, m6id, m6gd, m5, m5a, m5ad, m5n, m5dn, m5d, m5zn, m4
Mémoire optimisée (R)	r8g, r7a, r7iz, r7d, r7g, r6a, r6i, r6id, r6in, r6idn, r6g, r6gd, r5, r5n, r5a, r5dn, r5b, r5ad, r5d, r4
Éclatable (T)	t4g, t3, t3a, t2
Mémoire élevée (Z/X)	z1d, x8g, x2gd
Stockage optimisé (I/D)	i8g, i7ie, i4g, i4i, i3, i3en, is4gen, d3, d3en, im4gn
Informatique accélérée (P/G/Inf/Trn)	p5, p4d, p4de, p3, p3dn, gr6, g6, g6e, g5g, g5, g4dn, inf2, inf1, trn1, trn1n

Famille	Types d'instances
Calcul haute performance (X2)	x2iezn, x2idn, x2idn

En outre, le mode automatique d'EKS ne créera que EC2 des instances répondant aux exigences suivantes :

- Plus d'un processeur
- La taille de l'instance n'est pas nanométrique, micro ou petite

Pour plus d'informations, consultez les conventions de dénomination des types d' EC2 instances Amazon.

Service des métadonnées d'instance

- Le mode automatique EKS est IMDSv2 appliqué avec une limite de saut de 1 par défaut, conformément aux meilleures pratiques AWS de sécurité.
- Cette configuration par défaut ne peut pas être modifiée en mode automatique.
- Pour les modules complémentaires qui nécessitent généralement un accès IMDS, fournissez des paramètres (tels que AWS la région) lors de l'installation afin d'éviter les recherches IMDS. Pour de plus amples informations, veuillez consulter the section called "Champs que vous pouvez personnaliser".
- Si un pod nécessite absolument un accès à l'IMDS lorsqu'il fonctionne en mode automatique, le pod doit être configuré pour fonctionner avec`hostNetwork: true`. Cela permet au Pod d'accéder directement au service de métadonnées de l'instance.
- Tenez compte des implications en matière de sécurité lorsque vous accordez à Pods l'accès aux métadonnées des instances.

Pour plus d'informations sur le service de métadonnées d' EC2 instance Amazon (IMDS), consultez Configurer les options du service de métadonnées d'instance dans le guide de l' EC2 utilisateur Amazon.

Considérations

- Si le stockage éphémère configuré dans le NodeClass est plus petit que le stockage NVMe local de l'instance, le mode automatique d'EKS élimine le besoin de configuration manuelle en effectuant automatiquement les actions suivantes :
  - Utilise un volume de données Amazon EBS inférieur (20 GiB) pour réduire les coûts.
  - Formate et configure le stockage NVMe local pour une utilisation éphémère des données. Cela inclut la configuration d'une matrice RAID 0 s'il existe plusieurs NVMe disques.
- Le mode automatique Amazon EKS ne prend pas en charge le service d'injection de AWS défauts. Pour plus d'informations, consultez la section Gestion des expériences du service d'injection de défauts dans le guide de l'utilisateur de AWS Resilience Hub.
- Il n'est pas nécessaire d'installer le `Neuron Device Plugin` sur les nœuds du mode automatique EKS.

  Si vous avez d'autres types de nœuds dans votre cluster, vous devez configurer le plug-in Neuron Device pour qu'il ne s'exécute pas sur les nœuds en mode automatique. Pour de plus amples informations, veuillez consulter the section called "Déploiement du contrôle".

En savoir plus sur l'identité et l'accès en mode automatique EKS

Cette rubrique décrit les rôles et les autorisations requis pour utiliser le mode automatique d'EKS (Identity and Access Management) (IAM). Le mode automatique d'EKS utilise deux rôles IAM principaux : un rôle IAM de cluster et un rôle IAM de nœud. Ces rôles fonctionnent conjointement avec EKS Pod Identity et les entrées d'accès EKS pour fournir une gestion complète des accès pour vos clusters EKS.

Lorsque vous configurez le mode automatique d'EKS, vous devez configurer ces rôles IAM avec des autorisations spécifiques qui permettent aux AWS services d'interagir avec les ressources de votre cluster. Cela inclut les autorisations pour gérer les ressources informatiques, les volumes de stockage, les équilibreurs de charge et les composants réseau. Il est essentiel de comprendre ces configurations de rôles pour garantir le bon fonctionnement et la sécurité du cluster.

En mode automatique d'EKS, les rôles AWS IAM sont automatiquement mappés aux autorisations Kubernetes via les entrées d'accès EKS, ce qui élimine le besoin de configuration manuelle ou de liaisons personnalisées. `aws-auth` ConfigMaps Lorsque vous créez un nouveau cluster en mode automatique, EKS crée automatiquement les autorisations Kubernetes correspondantes à l'aide des entrées Access, garantissant ainsi que les AWS services et les composants du cluster disposent

des niveaux d'accès appropriés au sein du système d'autorisation AWS et du système d'autorisation Kubernetes. Cette intégration automatisée réduit la complexité de configuration et permet d'éviter les problèmes liés aux autorisations qui se produisent fréquemment lors de la gestion des clusters EKS.

Rôle IAM du cluster

Le rôle Cluster IAM est un rôle AWS Identity and Access Management (IAM) utilisé par Amazon EKS pour gérer les autorisations pour les clusters Kubernetes. Ce rôle accorde à Amazon EKS les autorisations nécessaires pour interagir avec d'autres AWS services au nom de votre cluster et est automatiquement configuré avec des autorisations Kubernetes à l'aide des entrées d'accès EKS.

- Vous devez associer des politiques AWS IAM à ce rôle.
- Le mode automatique EKS associe automatiquement les autorisations Kubernetes à ce rôle à l'aide des entrées d'accès EKS.
- Avec le mode automatique d'EKS, AWS suggère de créer un seul rôle IAM de cluster par AWS compte.
- AWS suggère de nommer ce rôle`AmazonEKSAutoClusterRole`.
- Ce rôle nécessite des autorisations pour plusieurs AWS services afin de gérer les ressources, notamment les volumes EBS, les Elastic Load Balancers et EC2 les instances.
- La configuration suggérée pour ce rôle inclut plusieurs politiques IAM AWS gérées, liées aux différentes fonctionnalités du mode automatique d'EKS.
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`

Pour plus d'informations sur le rôle IAM du cluster et les politiques IAM AWS gérées, voir :

- the section called “ AWS politiques gérées”
- the section called “Rôle IAM du cluster”

Pour plus d'informations sur l'accès à Kubernetes, consultez :

- the section called “Passez en revue les politiques d'accès”

Rôle IAM de nœud

Le rôle Node IAM est un rôle AWS Identity and Access Management (IAM) utilisé par Amazon EKS pour gérer les autorisations des nœuds de travail dans les clusters Kubernetes. Ce rôle accorde aux EC2 instances exécutées en tant que nœuds Kubernetes les autorisations nécessaires pour interagir avec les AWS services et les ressources, et est automatiquement configuré avec les autorisations RBAC Kubernetes à l'aide des entrées d'accès EKS.

- Vous devez associer des politiques AWS IAM à ce rôle.
- Le mode automatique EKS associe automatiquement les autorisations RBAC Kubernetes à ce rôle à l'aide des entrées d'accès EKS.
- AWS suggère de nommer ce rôle`AmazonEKSAutoNodeRole`.
- Avec le mode automatique d'EKS, AWS suggère de créer un rôle IAM de nœud unique par AWS compte.
- Ce rôle possède des autorisations limitées. Les principales autorisations incluent le fait d'assumer un rôle d'identité de module et d'extraire des images de l'ECR.
- AWS suggère les politiques IAM AWS gérées suivantes :
  - `AmazonEKSWorkerNodeMinimalPolicy`
  - `AmazonEC2ContainerRegistryPullOnly`

Pour plus d'informations sur le rôle IAM du cluster et les politiques IAM AWS gérées, voir :

- the section called " AWS politiques gérées"
- the section called "Rôle IAM de nœud"

Pour plus d'informations sur l'accès à Kubernetes, consultez :

- the section called "Passez en revue les politiques d'accès"

Rôle lié à un service

Amazon EKS utilise un rôle lié à un service (SLR) pour certaines opérations. Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés au service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

AWS crée et configure automatiquement le reflex. Vous ne pouvez supprimer un appareil photo reflex qu'après avoir d'abord supprimé les ressources associées. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

La politique SLR accorde à Amazon EKS l'autorisation d'observer et de supprimer les principaux composants de l'infrastructure : EC2 ressources (instances, interfaces réseau, groupes de sécurité), ressources ELB (équilibreurs de charge, groupes cibles), CloudWatch fonctionnalités (journalisation et métriques) et rôles IAM avec le préfixe « eks ». Il permet également la mise en réseau de terminaux privés via l'association de VPC/hosted zones et inclut des autorisations pour la EventBridge surveillance et le nettoyage des ressources étiquetées EKS.

Pour plus d'informations, consultez :

- the section called "AWS politique gérée : Amazon EKSService RolePolicy"
- the section called "Autorisations du rôle lié à un service pour Amazon EKS"

AWS Balises personnalisées pour les ressources EKS Auto

Par défaut, les politiques gérées liées au mode automatique d'EKS n'autorisent pas l'application de balises définies par l'utilisateur aux AWS ressources provisionnées en mode automatique. Si vous souhaitez appliquer des balises définies par l'utilisateur aux AWS ressources, vous devez attribuer des autorisations supplémentaires au rôle IAM du cluster avec des autorisations suffisantes pour créer et modifier des balises sur les AWS ressources. Vous trouverez ci-dessous un exemple de politique qui autorisera un accès illimité au balisage :

Afficher un exemple de politique de balise personnalisée

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Compute",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateFleet",
                "ec2:RunInstances",
                "ec2:CreateLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-node-class-name": "*",
                    "aws:RequestTag/eks:kubernetes-node-pool-name": "*"
                }
            }
        },
        {
            "Sid": "Storage",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVolume",
                "ec2:CreateSnapshot"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:volume/*",
                "arn:aws: ec2:*:*:snapshot/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "Networking",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-cni-node-name": "*"
                }
            }
        },
        {
```

```
            "Sid": "LoadBalancer",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateRule",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldProtection",
            "Effect": "Allow",
            "Action": [
                "shield:CreateProtection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldTagResource",
            "Effect": "Allow",
            "Action": [
                "shield:TagResource"
            ],
            "Resource": "arn:aws: shield::*:protection/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
```

```
        }
    ]
}
```

Référence de politique d'accès

Pour plus d'informations sur les autorisations Kubernetes utilisées par le mode automatique d'EKS, consultez. the section called “Passez en revue les politiques d'accès”

En savoir plus sur la mise en réseau VPC et l'équilibrage de charge en mode automatique EKS

Cette rubrique explique comment configurer les fonctionnalités de mise en réseau et d'équilibrage de charge du Virtual Private Cloud (VPC) en mode automatique EKS. Bien que le mode automatique d'EKS gère automatiquement la plupart des composants réseau, vous pouvez toujours personnaliser certains aspects de la configuration réseau de votre cluster grâce aux `NodeClass` ressources et aux annotations de l'équilibreur de charge.

Lorsque vous utilisez le mode automatique EKS, il AWS gère la configuration de l'interface réseau de conteneurs VPC (CNI) et le provisionnement de l'équilibreur de charge pour votre cluster. Vous pouvez influencer les comportements du réseau en définissant `NodeClass` des objets et en appliquant des annotations spécifiques à vos ressources de service et d'entrée, tout en conservant le modèle opérationnel automatisé fourni par le mode automatique d'EKS.

Capacité de mise en réseau

Le mode automatique EKS dispose d'une nouvelle fonctionnalité réseau qui gère la mise en réseau des nœuds et des pods. Vous pouvez le configurer en créant un objet `NodeClass` Kubernetes.

Les options de configuration du AWS VPC CNI précédent ne s'appliqueront pas au mode automatique EKS.

Configurer le réseau à l'aide d'un **`NodeClass`**

La `NodeClass` ressource en mode automatique d'EKS vous permet de personnaliser certains aspects de la capacité réseau. Vous pouvez ainsi spécifier les sélections de groupes de sécurité, contrôler le placement des nœuds sur les sous-réseaux VPC, définir des politiques SNAT, configurer des politiques réseau et activer la journalisation des événements réseau. `NodeClass` Cette approche maintient le modèle opérationnel automatisé du mode automatique d'EKS tout en offrant la flexibilité nécessaire à la personnalisation du réseau.

Vous pouvez utiliser un `NodeClass` pour :

- Sélectionnez un groupe de sécurité pour les nœuds
- Contrôlez la manière dont les nœuds sont placés sur les sous-réseaux VPC
- Définissez la politique SNAT du nœud sur ou `random` `disabled`
- Activez les politiques réseau Kubernetes, notamment :
  - Définissez la politique réseau sur Refuser ou Autoriser par défaut
  - Activez la journalisation des événements réseau dans un fichier.
- Isolez le trafic des pods du trafic des nœuds en attachant des pods à différents sous-réseaux.

Découvrez comment créer un Amazon EKS NodeClass.

Considérations

Le mode automatique EKS prend en charge :

- Politiques du réseau EKS.
- Les `HostNetwork` options `HostPort` et pour les pods Kubernetes.
- Nœuds et pods dans des sous-réseaux publics ou privés.
- Mise en cache des requêtes DNS sur le nœud.

Le mode automatique EKS ne prend pas en charge :

- Groupes de sécurité par pod (SGPP).
- Mise en réseau personnalisée dans le`ENIConfig`. Vous pouvez placer des pods dans plusieurs sous-réseaux ou les isoler exclusivement du trafic des nœuds avecthe section called "Sélection de sous-réseaux pour les pods".
- Configurations Warm IP, Warm Prefix et Warm ENI.
- Configuration minimale des cibles IP.
- Activation ou désactivation de la délégation de préfixes.
- Autres configurations prises en charge par le AWS VPC CNI open source.
- Configurations de politique réseau telles que la personnalisation du temporisateur Conntrack (la valeur par défaut est 300 s).
- Exportation des journaux d'événements réseau vers CloudWatch.

Gestion des ressources du réseau

Le mode automatique EKS gère les préfixes, l'adressage IP et la gestion de l'interface réseau en surveillant les NodeClass ressources pour les configurations réseau. Le service effectue automatiquement plusieurs opérations clés :

Délégation de préfixes

Le mode automatique d'EKS fournit des `/28` IPv4 préfixes à l'interface réseau principale pour les nœuds et gère un pool de ressources prédéfini qui évolue en fonction du nombre de pods planifiés. Si nécessaire, il fournit des interfaces réseau secondaires avec des groupes de sécurité identiques à ceux de l'interface principale du sous-réseau du nœud. Si les préfixes ne sont plus disponibles dans le sous-réseau, le service revient aux adresses secondaires IPv4 .

Gestion du temps de recharge

Le service implémente un pool de recharge pour les préfixes ou IPv4 les adresses secondaires qui ne sont plus utilisés. Une fois le délai de recharge expiré, ces ressources sont renvoyées au VPC. Toutefois, si les pods réutilisent ces ressources pendant la période de recharge, elles sont restaurées à partir du pool de recharge.

IPv6 Support

Pour les IPv6 clusters, le mode automatique EKS fournit un `/80` IPv6 préfixe par nœud sur l'interface réseau principale.

Le service assure également la bonne gestion et le ramassage des déchets de toutes les interfaces réseau.

Equilibrage de charge

Vous configurez les équilibreurs de charge AWS élastiques fournis par EKS Auto Mode à l'aide d'annotations sur les ressources de service et d'entrée.

Pour plus d'informations, consultez the section called "Créer une classe d'entrée" ou the section called "Créer un service".

Considérations relatives à l'équilibrage de charge avec le mode automatique EKS

- Le mode de ciblage par défaut est le mode IP et non le mode instance.
- Le mode automatique EKS prend uniquement en charge le mode groupe de sécurité pour les équilibreurs de charge réseau.

- AWS ne prend pas en charge la migration des équilibreurs de charge depuis le contrôleur d'équilibrage de AWS charge autogéré vers le mode automatique EKS.
- Le `networking.ingress.ipBlock` champ indiqué dans les `TargetGroupBinding` spécifications n'est pas pris en charge.
- Si vos nœuds de travail utilisent des groupes de sécurité personnalisés (et non un modèle de `eks-cluster-sg-` dénomination), votre rôle de cluster nécessite des autorisations IAM supplémentaires. La politique gérée par EKS par défaut permet uniquement à EKS de modifier les groupes de sécurité nommés. `eks-cluster-sg-` Sans l'autorisation de modifier vos groupes de sécurité personnalisés, EKS ne peut pas ajouter les règles d'entrée requises permettant au trafic ALB/NLB d'atteindre vos pods.

Résoudre les problèmes liés au mode automatique d'EKS

Avec le mode automatique d'EKS, vous AWS assumez une plus grande responsabilité pour les EC2 instances de votre AWS compte. EKS assume la responsabilité de l'exécution du conteneur sur les nœuds, du système d'exploitation sur les nœuds et de certains contrôleurs. Cela inclut un contrôleur de stockage par blocs, un contrôleur d'équilibrage de charge et un contrôleur de calcul.

Vous devez utiliser AWS Kubernetes APIs pour dépanner les nœuds. Vous pouvez :

- Utilisez une `NodeDiagnostic` ressource Kubernetes pour récupérer les journaux des nœuds à l'aide du. the section called "Agent de surveillance des nœuds" Pour plus d'étapes, voirthe section called "Obtenir les journaux des nœuds".
- Utilisez la commande AWS EC2 CLI `get-console-output` pour récupérer la sortie de console à partir des nœuds. Pour plus d'étapes, voirthe section called "Obtenez la sortie de console d'une instance EC2 gérée à l'aide de la AWS EC2 CLI".
- Utilisez les conteneurs de débogage Kubernetes pour récupérer les journaux des nœuds. Pour plus d'étapes, voirthe section called "Obtenez les journaux des nœuds à l'aide de conteneurs de débogage et de la CLI `kubectl`".

> **Note**
>
> Le mode automatique EKS utilise des instances EC2 gérées. Vous ne pouvez pas accéder directement aux instances EC2 gérées, y compris par SSH.

Vous pouvez rencontrer les problèmes suivants qui ont des solutions spécifiques aux composants du mode automatique d'EKS :

- Pods bloqués dans `Pending` cet état, qui ne sont pas planifiés sur les nœuds du mode automatique. Pour les solutions, voirthe section called "Résoudre les problèmes liés à l'échec de la planification du Pod sur le nœud en mode automatique".
- EC2 instances gérées qui ne rejoignent pas le cluster en tant que nœuds Kubernetes. Pour les solutions, voirthe section called "Résoudre les problèmes liés au fait que le nœud ne rejoint pas le cluster".
- Erreurs et problèmes liés au `NodePoolsPersistentVolumes`, et `Services` qui utilisent les contrôleurs inclus dans le mode automatique EKS. Pour les solutions, voirthe section called "Résoudre les problèmes liés aux manettes incluses en mode automatique".
- La sécurité améliorée des pods empêche le partage de volumes entre les pods. Pour les solutions, voirthe section called "Partage de volumes entre les pods".

Vous pouvez utiliser les méthodes suivantes pour dépanner les composants du mode automatique d'EKS :

- the section called "Obtenez la sortie de console d'une instance EC2 gérée à l'aide de la AWS EC2 CLI"
- the section called "Obtenez les journaux des nœuds à l'aide de conteneurs de débogage et de la CLI `kubectl`"
- the section called "Afficher les ressources associées au mode automatique EKS dans la AWS console"
- the section called "Afficher les erreurs IAM dans votre AWS compte"
- the section called "Détectez les problèmes de connectivité des nœuds avec `VPC Reachability Analyzer`"

Agent de surveillance des nœuds

Le mode automatique EKS inclut l'agent de surveillance des nœuds Amazon EKS. Vous pouvez utiliser cet agent pour consulter les informations de dépannage et de débogage relatives aux nœuds. L'agent de surveillance des nœuds publie Kubernetes `events` et le nœud. `conditions` Pour de plus amples informations, veuillez consulter the section called "Santé des nœuds".

Obtenez la sortie de console d'une instance EC2 gérée à l'aide de la AWS EC2 CLI

Cette procédure permet de résoudre les problèmes liés au démarrage ou au niveau du noyau.

Tout d'abord, vous devez déterminer l'ID d' EC2 instance de l'instance associée à votre charge de travail. Ensuite, utilisez la AWS CLI pour récupérer la sortie de la console.

1. Confirmez que vous avez `kubectl` installé votre cluster et que vous vous y êtes connecté
2. (Facultatif) Utilisez le nom d'un déploiement Kubernetes pour répertorier les pods associés.

```
kubectl get pods -l app=<deployment-name>
```

3. Utilisez le nom du Kubernetes Pod pour déterminer l'ID d' EC2 instance du nœud associé.

```
kubectl get pod <pod-name> -o wide
```

4. Utilisez l'ID d' EC2 instance pour récupérer la sortie de la console.

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

Obtenez les journaux des nœuds à l'aide de conteneurs de débogage et de la CLI **`kubectl`**

La méthode recommandée pour récupérer les journaux à partir d'un nœud en mode automatique EKS consiste à utiliser `NodeDiagnostic` des ressources. Pour ces étapes, voirthe section called "Obtenir les journaux des nœuds".

Cependant, vous pouvez diffuser des journaux en direct à partir d'une instance à l'aide de la `kubectl debug node` commande. Cette commande lance un nouveau Pod sur le nœud que vous souhaitez déboguer, que vous pouvez ensuite utiliser de manière interactive.

1. Lancez un conteneur de débogage. La commande suivante utilise `i-01234567890123456` l'ID d'instance du nœud, `-it` alloue a `tty` et attach `stdin` pour une utilisation interactive et utilise le `sysadmin` profil du fichier kubeconfig.

```
kubectl debug node/i-01234567890123456 -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023
```

L'exemple qui suit illustre un résultat.

```
Creating debugging pod node-debugger-i-01234567890123456-nxb9c with container
 debugger on node i-01234567890123456.
If you don't see a command prompt, try pressing enter.
bash-5.2#
```

2. Depuis le shell, vous pouvez maintenant installer celui `util-linux-core` qui fournit la `nsenter` commande. `nsenter`À utiliser pour entrer l'espace de noms de montage du PID 1 (`init`) sur l'hôte et exécuter la `journalctl` commande pour diffuser les journaux depuis : `kubelet`

```
yum install -y util-linux-core
nsenter -t 1 -m journalctl -f -u kubelet
```

Pour des raisons de sécurité, l'image du conteneur Amazon Linux n'installe pas beaucoup de fichiers binaires par défaut. Vous pouvez utiliser la `yum whatprovides` commande pour identifier le package qui doit être installé pour fournir un binaire donné.

```
yum whatprovides ps
```

```
Last metadata expiration check: 0:03:36 ago on Thu Jan 16 14:49:17 2025.
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : @System
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps

procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : amazonlinux
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps
```

Afficher les ressources associées au mode automatique EKS dans la AWS console

Vous pouvez utiliser la AWS console pour afficher l'état des ressources associées à votre cluster EKS Auto Mode.

- Volumes EBS
  - Afficher les volumes du mode automatique EKS en recherchant la clé du tag `eks:eks-cluster-name`
- Équilibreurs de charge
  - Afficher les équilibreurs de charge en mode automatique EKS en recherchant la clé du tag `eks:eks-cluster-name`
- EC2 Instances
  - Afficher les instances du mode automatique EKS en recherchant la clé du tag `eks:eks-cluster-name`

Afficher les erreurs IAM dans votre AWS compte

1. Accédez à la CloudTrail console
2. Sélectionnez « Historique des événements » dans le volet de navigation de gauche
3. Appliquez des filtres de code d'erreur :
   - AccessDenied
   - UnauthorizedOperation
   - InvalidClientTokenId

Recherchez les erreurs liées à votre cluster EKS. Utilisez les messages d'erreur pour mettre à jour vos entrées d'accès EKS, votre rôle IAM de cluster ou votre rôle IAM de nœud. Vous devrez peut-être associer une nouvelle politique à ces rôles avec des autorisations pour le mode automatique EKS.

Résoudre les problèmes liés à l'échec de la planification du Pod sur le nœud en mode automatique

Si les pods restent dans `Pending` cet état et ne sont pas programmés sur un nœud en mode auto, vérifiez si votre pod ou votre manifeste de déploiement possède un`nodeSelector`. Si un `nodeSelector` est présent, assurez-vous qu'il est utilisé `eks.amazonaws.com/compute-type: auto` pour être planifié sur les nœuds créés par le mode automatique EKS. Pour plus d'informations sur les étiquettes de nœuds utilisées par le mode automatique d'EKS, reportez-vous àthe section called “Déploiement du contrôle”.

Résoudre les problèmes liés au fait que le nœud ne rejoint pas le cluster

Le mode automatique d'EKS configure automatiquement les nouvelles EC2 instances avec les informations correctes pour rejoindre le cluster, notamment le point de terminaison du cluster et l'autorité de certification (CA) du cluster. Cependant, ces instances peuvent toujours ne pas rejoindre le cluster EKS en tant que nœud. Exécutez les commandes suivantes pour identifier les instances qui n'ont pas rejoint le cluster :

1. Courez `kubectl get nodeclaim` pour vérifier si `NodeClaims` c'est le cas`Ready = False`.

   ```
   kubectl get nodeclaim
   ```

2. Exécutez `kubectl describe nodeclaim <node_claim>` et examinez la section État pour détecter tout problème empêchant le nœud de rejoindre le cluster.

   ```
   kubectl describe nodeclaim <node_claim>
   ```

Messages d'erreur courants :

`Error getting launch template configs`

Cette erreur peut s'afficher si vous définissez des balises personnalisées `NodeClass` avec les autorisations de rôle IAM par défaut du cluster. Consultez the section called “Identité et accès”.

`Error creating fleet`

Il peut y avoir un problème d'autorisation lors de l'`RunInstances`appel depuis l' EC2 API. Vérifiez AWS CloudTrail les erreurs et the section called “Rôle IAM du cluster en mode automatique” les autorisations IAM requises.

Détectez les problèmes de connectivité des nœuds avec **`VPC Reachability Analyzer`**

> **Note**
>
> Vous êtes facturé pour chaque analyse exécutée par le VPC Reachability Analyzer. Pour plus de détails sur les tarifs, consultez la section Tarification Amazon VPC.

L'une des raisons pour lesquelles une instance n'a pas rejoint le cluster est un problème de connectivité réseau qui l'empêche d'atteindre le serveur d'API. Pour diagnostiquer ce problème, vous pouvez utiliser le VPC Reachability Analyzer pour effectuer une analyse de la connectivité entre un nœud qui ne parvient pas à rejoindre le cluster et le serveur d'API. Vous aurez besoin de deux informations :

- ID d'instance d'un nœud qui ne peut pas rejoindre le cluster
- Adresse IP du point de terminaison du serveur d'API Kubernetes

Pour obtenir l'ID d'instance, vous devez créer une charge de travail sur le cluster afin que le mode automatique EKS lance une EC2 instance. Cela crée également un `NodeClaim` objet dans votre cluster qui aura l'ID d'instance. Exécutez `kubectl get nodeclaim -o yaml` pour imprimer tous les éléments `NodeClaims` de votre cluster. Chacun `NodeClaim` contient l'ID d'instance sous forme de champ, puis de nouveau dans le ProviderID :

```
kubectl get nodeclaim -o yaml
```

L'exemple qui suit illustre un résultat.

```
    nodeName: i-01234567890123456
    providerID: aws:///us-west-2a/i-01234567890123456
```

Vous pouvez déterminer le point de terminaison de votre serveur d'API Kubernetes en exécutant. `kubectl get endpoint kubernetes -o yaml` Les adresses se trouvent dans le champ des adresses :

```
kubectl get endpoints kubernetes -o yaml
```

L'exemple qui suit illustre un résultat.

```
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes
  namespace: default
subsets:
- addresses:
  - ip: 10.0.143.233
  - ip: 10.0.152.17
```

```
  ports:
  - name: https
    port: 443
    protocol: TCP
```

Avec ces deux informations, vous pouvez effectuer l'analyse s. Accédez d'abord à l'Analyzer de Reachability VPC dans le. AWS Management Console

1. Cliquez sur « Créer et analyser le chemin »
2. Donnez un nom à l'analyse (par exemple « Node Join Failure »)
3. Pour le « Type de source », sélectionnez « Instances »
4. Entrez l'ID d'instance du nœud défaillant comme « Source »
5. Pour le « Path Destination », sélectionnez « Adresse IP »
6. Entrez l'une des adresses IP du serveur API comme « adresse de destination »
7. Développez la section « Configuration d'en-têtes de paquets supplémentaires »
8. Entrez un « port de destination » de 443
9. Sélectionnez « Protocol » comme TCP s'il n'est pas déjà sélectionné
10. Cliquez sur « Créer et analyser le chemin »
11. L'analyse peut prendre quelques minutes. Si les résultats de l'analyse indiquent un échec d'accessibilité, ils indiqueront l'emplacement de la défaillance sur le chemin réseau afin que vous puissiez résoudre le problème.

Partage de volumes entre les pods

Les nœuds en mode automatique EKS sont configurés SELinux en mode d'application, ce qui permet une meilleure isolation entre les pods exécutés sur le même nœud. Lorsque cette option SELinux est activée, la plupart des pods non privilégiés se verront automatiquement appliquer leur propre étiquette de sécurité multi-catégories (MCS). Cette étiquette MCS est unique par pod et est conçue pour garantir qu'un processus dans un pod ne peut pas manipuler un processus dans un autre pod ou sur l'hôte. Même si un pod étiqueté s'exécute en tant que root et a accès au système de fichiers hôte, il ne pourra pas manipuler les fichiers, effectuer des appels système sensibles sur l'hôte, accéder au runtime du conteneur ou obtenir la clé secrète de Kubelet.

De ce fait, vous pouvez rencontrer des problèmes lorsque vous essayez de partager des données entre des pods. Par exemple, un mode `PersistentVolumeClaim` d'accès de n'`ReadWriteOnce`autorisera toujours pas plusieurs pods à accéder simultanément au volume.

Pour activer ce partage entre les pods, vous pouvez utiliser les pods `seLinuxOptions` pour configurer la même étiquette MCS sur ces pods. Dans cet exemple, nous attribuons les trois catégories `c123,c456,c789` au Pod. Cela n'entrera pas en conflit avec les catégories attribuées automatiquement aux pods sur le nœud, car seules deux catégories leur seront attribuées.

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456,c789"
```

Afficher les événements Karpenter dans les journaux du plan de contrôle

Pour les clusters EKS sur lesquels les journaux du plan de contrôle sont activés, vous pouvez obtenir des informations sur les actions et le processus décisionnel de Karpenter en interrogeant les journaux. Cela peut être particulièrement utile pour résoudre les problèmes liés au mode automatique d'EKS liés au provisionnement, à la mise à l'échelle et à la terminaison des nœuds. Pour consulter les événements liés à Karpenter, utilisez la requête CloudWatch Logs Insights suivante :

```
fields @timestamp, @message
| filter @logStream like /kube-apiserver-audit/
| filter @message like 'DisruptionBlocked'
or @message like 'DisruptionLaunching'
or @message like 'DisruptionTerminating'
or @message like 'DisruptionWaitingReadiness'
or @message like 'Unconsolidatable'
or @message like 'FailedScheduling'
or @message like 'NoCompatibleInstanceTypes'
or @message like 'NodeRepairBlocked'
or @message like 'Disrupted'
or @message like 'Evicted'
or @message like 'FailedDraining'
or @message like 'TerminationGracePeriodExpiring'
or @message like 'TerminationFailed'
or @message like 'FailedConsistencyCheck'
or @message like 'InsufficientCapacityError'
or @message like 'UnregisteredTaintMissing'
or @message like 'NodeClassNotReady'
sort @timestamp desc
```

Cette requête filtre les événements spécifiques liés à Karpenter dans les journaux d'audit de kube-apiserver. Les événements incluent divers états de perturbation, des échecs de planification, des

problèmes de capacité et des problèmes liés aux nœuds. En analysant ces journaux, vous pouvez mieux comprendre :

- Pourquoi Karpenter prend certaines mesures.
- Tout problème empêchant le provisionnement, le dimensionnement ou la résiliation appropriés des nœuds.
- Problèmes potentiels de capacité ou de compatibilité avec les types d'instances.
- Événements liés au cycle de vie des nœuds, tels que les perturbations, les expulsions ou les résiliations.

Pour utiliser cette requête, procédez comme suit :

1. Accédez à la CloudWatch console
2. Sélectionnez « Logs Insights » dans le volet de navigation de gauche
3. Choisissez le groupe de journaux pour les journaux du plan de contrôle de votre cluster EKS
4. Collez la requête dans l'éditeur de requêtes
5. Ajustez la plage de temps selon les besoins
6. Exécuter la requête

Les résultats vous montreront une chronologie des événements liés à Karpenter, vous aidant à résoudre les problèmes et à comprendre le comportement du mode automatique EKS dans votre cluster. Pour examiner les actions de Karpenter sur un nœud spécifique, vous pouvez ajouter le filtre de ligne ci-dessous spécifiant l'ID d'instance à la requête susmentionnée :

```
|filter @message like /[.replaceable]`i-12345678910123456`/
```

> **Note**
>
> Pour utiliser cette requête, la journalisation du plan de contrôle doit être activée sur votre cluster EKS. Si vous ne l'avez pas encore fait, veuillez vous référer àthe section called "Journaux du plan de contrôle".

Résoudre les problèmes liés aux manettes incluses en mode automatique

Si vous rencontrez un problème avec une manette, vous devez rechercher :

- Si les ressources associées à ce contrôleur sont correctement formatées et valides.
- Si les ressources AWS IAM et Kubernetes RBAC sont correctement configurées pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Identité et accès".

Consultez les notes de mise à jour d'EKS Auto Mode

Cette page décrit les mises à jour apportées au mode automatique d'Amazon EKS. Vous pouvez consulter régulièrement cette page pour les annonces concernant les fonctionnalités, les corrections de bogues, les problèmes connus et les fonctionnalités obsolètes.

Pour recevoir des notifications concernant toutes les modifications apportées au fichier source sur cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante avec un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/
auto-change.adoc.atom
```

6 août 2025

Fonctionnalité : Ajout d'une nouvelle configuration NodeClass `spec.advancedNetworking.associatePublicIPAddress` qui peut être utilisée pour empêcher l'attribution d'adresses IP publiques aux nœuds en mode automatique EKS

30 juin 2025

Fonctionnalité : Le mode automatique utilise NodeClass désormais la clé KMS personnalisée configurée pour chiffrer le volume racine en lecture seule de l'instance, en plus du volume de données. read/write Auparavant, la clé KMS personnalisée était uniquement utilisée pour chiffrer le volume de données.

20 juin 2025

Fonctionnalité : Support pour contrôler le déploiement des charges de travail dans les réservations de capacité EC2 à la demande (ODCRs). Cela ajoute la clé facultative `capacityReservationSelectorTerms` au NodeClass, vous permettant de contrôler

explicitement les charges de travail utilisées par ODCRs vos charges de travail. Pour de plus amples informations, veuillez consulter the section called "Contrôlez le déploiement de l'ODCR".

13 juin 2025

Fonctionnalité : Support pour des sous-réseaux de pods distincts dans le`NodeClass`. Cela ajoute les clés facultatives `` `podSubnetSelectorTerms `` et permet `podSecurityGroupSelectorTerms` de définir les sous-réseaux et les groupes de sécurité pour les pods. Pour de plus amples informations, veuillez consulter the section called "Sélection de sous-réseaux pour les pods".

30 avril 2025

Fonctionnalité : Support pour les proxys réseau de transfert dans le`NodeClass`. Cela ajoute la clé facultative `advancedNetworking` pour configurer votre proxy HTTPS. Pour de plus amples informations, veuillez consulter the section called "Spécification de la classe de nœuds".

18 avril 2025

Fonctionnalité : Support pour la résolution de domaines .local (généralement réservés au DNS multicast) via le DNS unicast.

11 avril 2025

Fonctionnalité : Ajouté `certificateBundles` et `ephemeralStorage.kmsKeyID` à`NodeClass`. Pour de plus amples informations, veuillez consulter the section called "Spécification de la classe de nœuds".

Fonctionnalité : vitesse d'extraction des images améliorée, en particulier pour les types d'instance dotés d'un stockage d'instance local qui peuvent tirer parti de la décompression plus rapide des images.

Correction de bogue : résolution d'un problème de course à l'origine FailedCreatePodSandBox d'une erreur lors de la numérotation : dial tcp 127.0.0. 1:50051 : connexion : la connexion était parfois refusée pour les pods planifiés vers un nœud immédiatement au démarrage.

4 avril 2025

Fonctionnalité : augmentation `registryPullQPS` de 5 à 25 et `registryBurst` de 10 à 50 pour réduire la limitation d'attraction des images imposée par le client () `Failed to pull image xyz: pull QPS exceeded`

31 mars 2025

Correction de bogue : correction d'un problème selon lequel, si un module DNS principal était exécuté sur un nœud en mode automatique, les requêtes DNS provenant des pods du nœud arrivaient sur ce pod DNS principal plutôt que sur le serveur DNS local du nœud. Les requêtes DNS provenant de Pods sur un nœud en mode automatique seront toujours envoyées au DNS local du nœud.

21 mars 2025

Correction de bogue : les nœuds en mode automatique se résolvent désormais `kube-dns.kube-system.svc.cluster.local` correctement lorsqu'aucun `kube-dns` service n'est installé dans le cluster. Résout GitHub le problème #2546.

14 mars 2025

Fonctionnalité : `IPv4` sortie activée dans les `IPv6` clusters. `IPv4`le trafic sortant des clusters en mode `IPv6` automatique sera désormais automatiquement traduit à l'`v4`adresse du nœud principal ENI.

Cycle de vie et configuration du cluster Amazon EKS

Cette section fournit des conseils détaillés sur la gestion complète du cycle de vie des clusters Kubernetes et sur les différentes manières de les configurer. Il couvre la création de clusters, la surveillance de l'état des clusters, la mise à jour des versions de Kubernetes et la suppression de clusters. Il inclut également des sujets avancés sur la façon de configurer l'accès aux terminaux, le support enable/disable Windows, la configuration de clusters privés, la mise en œuvre de l'autoscaling et la manière d'améliorer la résilience avec Zonal Shift pour la redirection du trafic dans les configurations multi-AZ.

Rubriques

- Création d'un cluster en mode automatique Amazon EKS
- Création d'un cluster Amazon EKS
- Préparez-vous aux mises à niveau des versions de Kubernetes et corrigez les erreurs de configuration grâce aux informations du cluster
- Mettre à jour le cluster existant vers la nouvelle version de Kubernetes
- Supprimer un cluster
- Point de terminaison du serveur API du cluster
- Déployer des nœuds Windows sur des clusters EKS
- Désactiver le support Windows
- Déployez des clusters privés avec un accès Internet limité
- Faites évoluer le calcul en cluster avec Karpenter et Cluster Autoscaler
- En savoir plus sur le changement de zone d'Amazon Application Recovery Controller (ARC) dans Amazon EKS
- Activez EKS Zonal Shift pour éviter les zones de disponibilité altérées

Création d'un cluster en mode automatique Amazon EKS

Cette rubrique fournit des instructions détaillées pour créer un cluster en mode automatique Amazon EKS à l'aide d'options de configuration avancées. Il couvre les prérequis, les options réseau et les configurations complémentaires. Le processus inclut la configuration des rôles IAM, la configuration des paramètres du cluster, la spécification des paramètres réseau et la sélection des modules

complémentaires. Les utilisateurs peuvent créer des clusters à l'aide de la CLI AWS Management Console ou de la AWS CLI, avec step-by-step des instructions fournies pour les deux méthodes.

Pour les utilisateurs recherchant un processus de configuration moins complexe, reportez-vous aux instructions suivantes pour des étapes simplifiées de création de clusters :

- the section called "CLI exsctl"
- the section called " AWS CLI"
- the section called "Console de gestion"

Ce guide de configuration avancée est destiné aux utilisateurs qui ont besoin d'un contrôle plus précis de la configuration de leur cluster en mode automatique EKS et qui connaissent bien les concepts et les exigences d'Amazon EKS. Avant de procéder à la configuration avancée, assurez-vous de remplir toutes les conditions préalables et de bien comprendre les exigences en matière de réseau et d'IAM pour les clusters en mode automatique EKS.

Le mode automatique EKS nécessite des autorisations IAM supplémentaires. Pour plus d'informations, consultez :

- the section called "Rôles IAM pour les clusters en mode automatique EKS"
- the section called "Identité et accès"

> **Note**
>
> Si vous souhaitez créer un cluster sans le mode automatique EKS, consultezthe section called "Créer un cluster ".
> Cette rubrique traite de la configuration avancée. Si vous souhaitez commencer à utiliser le mode automatique EKS, consultezthe section called "Créer un cluster".

Prérequis

- Un VPC et des sous-réseaux existants qui répondent aux exigences Amazon EKS. Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de bien connaître les exigences du VPC et du sous-réseau. Si vous n'avez pas de VPC ni de sous-réseaux, vous pouvez les créer à l'aide d'un modèle fourni par Amazon EKS. AWS CloudFormation

- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version`. Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
- Un directeur IAM autorisé à créer et à modifier des ressources EKS et IAM.

Créer un cluster - AWS console

1. Ouvrez la console Amazon EKS.
2. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
3. Sous Options de configuration, sélectionnez Configuration personnalisée.
   - Cette rubrique traite de la configuration personnalisée. Pour plus d'informations sur la configuration rapide, consultezthe section called "Console de gestion".
4. Vérifiez que le mode Use EKS Auto est activé.
   - Cette rubrique traite de la création de clusters avec le mode automatique EKS. Pour plus d'informations sur la création de clusters sans le mode automatique EKS, consultezthe section called "Créer un cluster ".
5. Sur la page Configurer le cluster, renseignez les champs suivants :
   - Nom : nom de votre cluster. Le nom ne peut contenir que des caractères alphanumériques (en distinguant majuscules et minuscules), des traits d'union et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Rôle IAM du cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé pour permettre au plan de contrôle Kubernetes de gérer AWS les ressources en votre nom. Si vous n'avez pas encore créé de rôle Cluster IAM pour le mode automatique d'EKS, sélectionnez le bouton Créer un rôle recommandé pour créer le rôle avec les autorisations requises dans la console IAM.

- Version Kubernetes : version de Kubernetes à utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous avez besoin d'une version antérieure.
- Politique de mise à niveau : politique de version de Kubernetes que vous souhaitez définir pour votre cluster. Si vous souhaitez que votre cluster fonctionne uniquement sur une version de support standard, vous pouvez choisir Standard. Si vous souhaitez que votre cluster bénéficie d'un support étendu à la fin du support standard d'une version, vous pouvez choisir Extended. Si vous sélectionnez une version de Kubernetes actuellement en support étendu, vous ne pouvez pas sélectionner le support standard comme option.

6. Dans la section Calcul en mode automatique de la page de configuration du cluster, entrez les champs suivants :
   - Pools de nœuds : déterminez si vous souhaitez utiliser les pools de nœuds intégrés. Pour de plus amples informations, veuillez consulter the section called "Passez en revue les pools de nœuds intégrés".
   - Rôle Node IAM : si vous activez l'un des pools de nœuds intégrés, vous devez sélectionner un rôle Node IAM. Le mode automatique EKS attribuera ce rôle aux nouveaux nœuds. Vous ne pouvez pas modifier cette valeur une fois le cluster créé. Si vous n'avez pas encore créé de rôle Node IAM pour le mode automatique d'EKS, sélectionnez le bouton Créer un rôle recommandé pour créer le rôle avec les autorisations requises. Pour plus d'informations sur ce rôle, consultez the section called "Identité et accès".
7. Dans la section Accès au cluster de la page de configuration du cluster, entrez les champs suivants :
   - Accès administrateur au cluster Bootstrap : le créateur du cluster est automatiquement un administrateur Kubernetes. Si vous souhaitez désactiver cette option, sélectionnez Interdire l'accès de l'administrateur du cluster.
   - Mode d'authentification du cluster — Le mode automatique EKS nécessite des entrées d'accès EKS, le mode d'authentification de l'API EKS. Vous pouvez éventuellement activer le mode `ConfigMap` d'authentification en sélectionnant EKS API et ConfigMap.
8. Entrez les champs restants sur la page de configuration du cluster :
   - Chiffrement des secrets : (facultatif) choisissez d'activer le chiffrement des secrets Kubernetes à l'aide d'une clé KMS. Vous pouvez également activer cette option une fois que vous avez créé votre cluster. Avant d'activer cette fonctionnalité, assurez-vous de connaître les informations contenues dansthe section called "Activer le chiffrement secret".

- Changement de zone ARC — Le mode automatique EKS ne prend pas en charge le changement de zone Arc.
- Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ".

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

9. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :

- VPC : choisissez un VPC existant qui répond aux exigences relatives au VPC Amazon EKS pour y créer votre cluster. Avant de choisir un VPC, nous vous recommandons de connaître toutes les exigences et considérations relatives à. the section called "Exigences requises pour le VPC et les sous-réseaux" Vous ne pouvez pas modifier le VPC que vous souhaitez utiliser après la création du cluster. Si aucun VPCs n'est répertorié, vous devez d'abord en créer un. Pour de plus amples informations, veuillez consulter the section called "Création d'un VPC".
- Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Vous devez en sélectionner au moins deux.

  Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner des sous-réseaux, nous vous recommandons de connaître toutes les exigences et considérations relatives aux VPC et aux sous-réseaux Amazon EKS.

  Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.
- Choisissez la famille d'adresses IP du cluster : vous pouvez choisir l'une ou l'autre IPv4et IPv6.

  Kubernetes attribue des `IPv4` adresses aux pods et aux services, par défaut. Avant de décider d'utiliser la `IPv6` famille, assurez-vous de connaître toutes les considérations et exigences énoncées dans les exigencesthe section called "Exigences et considérations requises pour les sous-réseaux", the section called "Exigences de groupe de sécurité" les considérations et les rubriques relatives aux VPC. the section called "IPv6" Si vous choisissez la `IPv6` famille, vous

ne pouvez pas spécifier de plage d'adresses pour que Kubernetes puisse attribuer des adresses de `IPv6` service comme vous le pouvez pour la famille. `IPv4` Kubernetes attribue des adresses de service à partir de la plage d'adresses locale unique (). `fc00::/7`

- (Facultatif) Choisissez Configurer la plage d'adresses IP du service Kubernetes et spécifiez une gamme de services **`IPv4`**.

  La spécification de votre propre plage peut aider à éviter les conflits entre les services Kubernetes et les autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

  Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne le spécifiez pas, Kubernetes attribue des adresses IP de service à partir des blocs CIDR ou CIDR. `10.100.0.0/16` `172.20.0.0/16`

- Pour Accès au point de terminaison de cluster, sélectionnez une option. Une fois votre cluster créé, vous pouvez modifier cette option. Avant de sélectionner une option autre que par défaut, assurez-vous de vous familiariser avec les options et leurs implications. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

10. (Facultatif) Sur la page Configurer l'observabilité, choisissez quelles options de Métriques et de Journalisation du plan de contrôle doivent être activées. Par défaut, chaque type de journal est désactivé.

    - Pour plus d'informations sur l'option Prometheus metrics, consultez. the section called "Étape 1 : activer les métriques Prometheus"
    - Pour plus d'informations sur les options de Journalisation du plan de contrôle, consultez the section called "Journaux du plan de contrôle".
    - Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

11. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster. Vous pouvez choisir autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que

vous le souhaitez. Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez cliquer sur la numérotation des pages pour afficher des résultats supplémentaires ou rechercher les modules complémentaires AWS Marketplace disponibles en saisissant du texte dans le champ de recherche. Vous pouvez également filtrer par catégorie, fournisseur ou modèle de tarification, puis choisir les modules complémentaires dans les résultats de recherche. Lorsque vous créez un cluster, vous pouvez afficher, sélectionner et installer tout module complémentaire compatible avec EKS Pod Identities, comme indiqué dansthe section called "Identité du pod".

- Le mode automatique d'EKS automatise les fonctionnalités de certains modules complémentaires. Si vous prévoyez de déployer des groupes de nœuds gérés par EKS sur votre cluster en mode automatique EKS, sélectionnez Add-ons Amazon EKS supplémentaires et passez en revue les options. Vous devrez peut-être installer des modules complémentaires tels que CoreDNS et kube-proxy. EKS installera les modules complémentaires de cette section uniquement sur des nœuds et des groupes de nœuds autogérés.
- Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

12. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez la version que vous voulez installer. Vous pourrez toujours effectuer une mise à jour vers une version ultérieure après la création du cluster.

    Pour les modules complémentaires compatibles avec EKS Pod Identities, vous pouvez utiliser la console pour générer automatiquement le rôle avec le nom, la politique AWS gérée et la politique de confiance préremplis spécifiquement pour le module complémentaire. Vous pouvez réutiliser les rôles existants ou créer de nouveaux rôles pour les modules complémentaires pris en charge. Pour les étapes d'utilisation de la console afin de créer des rôles pour les modules complémentaires compatibles avec EKS Pod Identities, voirthe section called "Créer un module complémentaire (AWS console)". Si un module complémentaire ne prend pas en charge EKS Pod Identity, un message contenant les instructions d'utilisation de l'assistant pour créer les rôles IAM pour les comptes de service (IRSA) s'affiche une fois le cluster créé.

    Vous pourrez mettre à jour la configuration de chaque module complémentaire après la création du cluster. Pour plus d'informations sur la configuration des modules complémentaires, consultez the section called "Mettre à jour un module complémentaire". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

13. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez

Modifier. Lorsque vous êtes satisfait, choisissez Create. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné.

> **Note**
>
> Vous pouvez recevoir un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

L'approvisionnement de cluster dure plusieurs minutes.

Créer un cluster - AWS CLI

Les instructions de la CLI suivantes concernent la création de ressources IAM et la création du cluster.

Création d'un rôle IAM de cluster en mode automatique EKS

Étape 1 : Création de la politique de confiance

Créez une politique de confiance qui autorise le service Amazon EKS à assumer ce rôle. Enregistrez la politique sous la forme suivante `trust-policy.json` :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
```

```
    }
  ]
}
```

Étape 2 : Création du rôle IAM

Utilisez la politique de confiance pour créer le rôle IAM du cluster :

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Étape 3 : Notez l'ARN du rôle

Récupérez et enregistrez l'ARN du nouveau rôle pour l'utiliser lors des étapes suivantes :

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Étape 4 : Joindre les politiques requises

Associez les politiques AWS gérées suivantes au rôle IAM du cluster pour accorder les autorisations nécessaires :

EKSClusterPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputePolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBlock StoragePolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
```

```
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

Amazon EKSLoad BalancingPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Création d'un rôle IAM pour un nœud en mode automatique EKS

Étape 1 : Création de la politique de confiance

Créez une politique de confiance qui autorise le service Amazon EKS à assumer ce rôle. Enregistrez la politique sous la forme suivante `node-trust-policy.json` :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Étape 2 : créer le rôle Node IAM

Utilisez le fichier node-trust-policy.json de l'étape précédente pour définir les entités qui peuvent assumer le rôle. Exécutez la commande suivante pour créer le rôle Node IAM :

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
```

```
    --assume-role-policy-document file://node-trust-policy.json
```

Étape 3 : Notez l'ARN du rôle

Après avoir créé le rôle, récupérez et enregistrez l'ARN du rôle Node IAM. Vous aurez besoin de cet ARN dans les étapes suivantes. Utilisez la commande suivante pour obtenir l'ARN :

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Étape 4 : Joindre les politiques requises

Associez les politiques AWS gérées suivantes au rôle Node IAM pour fournir les autorisations nécessaires :

Amazon EKSWorker NodeMinimalPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Amazon EC2 ContainerRegistryPullOnly :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Créer un cluster

1. Créez votre cluster à l'aide de la commande suivante. Avant d'exécuter la commande, effectuez les remplacements suivants :
   - *region-code*Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
   - Remplacez *my-cluster* par un nom pour votre cluster. Le nom ne peut contenir que des caractères alphanumériques (en distinguant majuscules et minuscules), des traits d'union et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - *1.30*Remplacez-le par n'importe quel eks/latest/userguide/kubernetes fichier -versions.html [version prise en charge par Amazon EKS, type="documentation "].
   - Remplacez *111122223333* par votre identifiant de compte

- Si vous avez créé des rôles IAM nommés différemment pour les rôles de cluster et de nœud, remplacez le ARNs.
- Remplacez les valeurs de `subnetIds` par vos propres valeurs. Vous pouvez également en ajouter d'autres IDs. Vous devez spécifier au moins deux sous-réseaux IDs.

  Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner des sous-réseaux, nous vous recommandons de connaître toutes les exigences et considérations relatives aux VPC et aux sous-réseaux Amazon EKS.
- Si vous ne souhaitez pas spécifier d'ID de groupe de sécurité, `,securityGroupIds=sg-<ExampleID1>` supprimez-le de la commande. Si vous souhaitez spécifier un ou plusieurs groupes de sécurité IDs, remplacez les valeurs de par `securityGroupIds` les vôtres. Vous pouvez également en ajouter d'autres IDs.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.

```
aws eks create-cluster \
  --region region-code \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --role-arn arn:aws: iam::111122223333:role/AmazonEKSAutoClusterRole \
  --resources-vpc-config '{"subnetIds": ["subnet-ExampleID1","subnet-ExampleID2"],
 "securityGroupIds": ["sg-ExampleID1"], "endpointPublicAccess": true,
 "endpointPrivateAccess": true}' \
  --compute-config '{"enabled": true, "nodeRoleArn": "arn:aws:
 iam::111122223333:role/AmazonEKSAutoNodeRole", "nodePools": ["general-purpose",
 "system"]}' \
  --kubernetes-network-config '{"elasticLoadBalancing": {"enabled": true}}' \
  --storage-config '{"blockStorage": {"enabled": true}}' \
  --access-config '{"authenticationMode": "API"}'
```

> **Note**
>
> Vous pouvez recevoir un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

Les paramètres suivants sont des paramètres facultatifs qui, si nécessaire, doivent être ajoutés à la commande précédente. Vous pouvez activer ces options uniquement lorsque vous créez le cluster, et non après.

- Si vous souhaitez spécifier à partir de quel bloc de routage inter-domaines sans classe (CIDR) `IPv4` Kubernetes attribue des adresses IP de service, vous devez le spécifier en ajoutant le `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` à la commande suivante.

  La spécification de votre propre plage peut aider à éviter les conflits entre les services Kubernetes et les autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

    Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne le spécifiez pas, Kubernetes attribue des adresses IP de service à partir des blocs CIDR ou CIDR. `10.100.0.0/16` `172.20.0.0/16`

- Si vous créez un cluster et que vous souhaitez que le cluster attribue des `IPv6` adresses aux pods et aux services au lieu d'`IPv4`adresses, ajoutez `--kubernetes-network-config ipFamily=ipv6` la commande suivante.

Kubernetes attribue des `IPv4` adresses aux pods et aux services, par défaut. Avant de décider d'utiliser la `IPv6` famille, assurez-vous de connaître toutes les considérations et exigences énoncées dans les exigencesthe section called "Exigences et considérations requises pour les sous-réseaux", the section called "Exigences de groupe de sécurité" les considérations et les rubriques relatives aux VPC. the section called "IPv6" Si vous choisissez la `IPv6` famille, vous ne pouvez pas spécifier de plage d'adresses pour que Kubernetes puisse attribuer des adresses de `IPv6` service comme vous le pouvez pour la famille. `IPv4` Kubernetes attribue des adresses de service à partir de la plage d'adresses locale unique (). `fc00::/7`

2. Le provisionnement du cluster prend quelques minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

Étapes suivantes

- the section called "Cluster d'accès avec kubectl"
- the section called "Accorder des autorisations"
- Activer le chiffrement des secrets pour votre cluster.
- Configurer la journalisation de votre cluster.
- Ajouter des nœuds à votre cluster.

Création d'un cluster Amazon EKS

> **Note**
>
> Cette rubrique traite de la création de clusters EKS sans le mode automatique EKS.
> Pour obtenir des instructions détaillées sur la création d'un cluster en mode automatique EKS, consultezthe section called "Créer un cluster automatique".
> Pour commencer à utiliser le mode automatique EKS, voirthe section called "Créer un cluster (mode automatique EKS)".

Cette rubrique fournit une vue d'ensemble des options disponibles et décrit les éléments à prendre en compte lorsque vous créez un cluster Amazon EKS. Si vous devez créer un cluster avec votre infrastructure sur site pour le calcul des nœuds, consultezthe section called "Créer un cluster". Si c'est la première fois que vous créez un cluster Amazon EKS, nous vous recommandons de suivre l'un de nos guidesMise en route. Ces guides vous aident à créer un cluster par défaut simple sans développer toutes les options disponibles.

Prérequis

- Un VPC et des sous-réseaux existants qui répondent aux exigences Amazon EKS. Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de bien connaître les exigences du VPC et du sous-réseau. Si vous n'avez pas de VPC ni de sous-réseaux, vous pouvez les créer à l'aide d'un modèle fourni par Amazon EKS. AWS CloudFormation
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un principal IAM disposant des autorisations `create` et `describe` sur un cluster Amazon EKS. Pour plus d'informations, consultez the section called "Création d'un cluster Kubernetes local sur un Outpost" et the section called "Affichage de la liste ou description de tous les clusters".

Étape 1 : créer le rôle IAM du cluster

1. Si vous possédez déjà un rôle IAM dans le cluster, ou si vous comptez créer votre cluster avec`eksctl`, vous pouvez ignorer cette étape. Par défaut, `eksctl` crée un rôle pour vous.
2. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

3. Créez le rôle IAM de cluster Amazon EKS. Si nécessaire, faites précéder *`eks-cluster-role-trust-policy.json`* par le chemin d'accès sur votre ordinateur sur lequel vous avez enregistré le fichier lors de l'étape précédente. La commande associe la politique d'approbation que vous avez créée lors de l'étape précédente à ce rôle. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

4. Vous pouvez attribuer la politique gérée par Amazon EKS ou créer votre propre politique personnalisée. Pour connaître les autorisations minimales que vous devez utiliser dans votre politique personnalisée, consultez the section called "Rôle IAM du cluster".

   Associez la politique gérée par Amazon EKS nommée Amazon EKSCluster Policy au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

Rôle lié au service

Amazon EKS crée automatiquement un rôle lié à un service appelé`AWSServiceRoleForAmazonEKS`.

Cela s'ajoute au rôle IAM du cluster. Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Ce rôle permet à Amazon EKS de gérer les clusters de votre compte. Pour de plus amples informations, veuillez consulter the section called "Rôle de cluster".

L'identité IAM que vous utilisez pour créer le cluster EKS doit être autorisée à créer le rôle lié à un service. Cela inclut l'`iam:CreateServiceLinkedRole`autorisation.

Si le rôle lié au service n'existe pas déjà et que votre rôle IAM actuel ne dispose pas des autorisations suffisantes pour le créer, l'opération de création du cluster échouera.

Étape 2 : créer un cluster

Vous pouvez créer un cluster en utilisant :

- `eksctl`
- le AWS Management Console
- la AWS CLI

Créer un cluster - eksctl

1. Vous avez besoin d'`0.210.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Créez un `IPv4` cluster Amazon EKS avec la version par défaut d'Amazon EKS Kubernetes dans votre région par défaut. AWS Avant d'exécuter la commande, effectuez les remplacements suivants :
3. *`region-code`*Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
4. Remplacez *`my-cluster`* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
5. *`1.33`*Remplacez-le par n'importe quel eks/latest/userguide/kubernetes fichier -versions.html [version prise en charge par Amazon EKS, type="documentation "].

6. Modifiez les valeurs de `vpc-private-subnets` pour répondre à vos besoins. Vous pouvez également en ajouter d'autres IDs. Vous devez spécifier au moins deux sous-réseaux IDs. Si vous préférez spécifier des sous-réseaux publics, vous pouvez passer `--vpc-private-subnets` à `--vpc-public-subnets`. Les sous-réseaux publics ont une table de routage associée avec une route vers une passerelle Internet, mais les sous-réseaux privés n'ont pas de table de routage associée. Nous recommandons d'utiliser des sous-réseaux privés dans la mesure du possible.

   Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner des sous-réseaux, nous vous recommandons de connaître toutes les exigences et considérations relatives aux VPC et aux sous-réseaux Amazon EKS.

7. Exécutez la commande suivante :

```
eksctl create cluster --name my-cluster --region region-code --version 1.33 --vpc-
private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

   L'approvisionnement de cluster dure plusieurs minutes. Pendant la création du cluster, plusieurs lignes de sortie apparaissent. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

8. Continuez avec the section called "Étape 3 : Mettre à jour kubeconfig"

Paramètres facultatifs

Pour afficher la plupart des options qui peuvent être spécifiées lors de la création d'un cluster avec `eksctl`, utilisez la commande `eksctl create cluster --help`. Pour consulter toutes les options disponibles, vous pouvez utiliser un fichier `config`. Pour plus d'informations, consultez Utilisation des fichiers de configuration et du schéma du fichier de configuration dans la documentation `eksctl`. Vous pouvez trouver des exemples de fichiers de configuration sur GitHub.

Les paramètres suivants sont des paramètres facultatifs qui, si nécessaire, doivent être ajoutés à la commande précédente. Vous pouvez activer ces options uniquement lorsque vous créez le cluster, et non après. Si vous devez spécifier ces options, vous devez créer le cluster avec un fichier de configuration eksctl et spécifier les paramètres, plutôt que d'utiliser la commande précédente.

- Si vous souhaitez spécifier un ou plusieurs groupes de sécurité qu'Amazon EKS attribue aux interfaces réseau qu'il crée, spécifiez l'option SecurityGroup.

Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.

- Si vous souhaitez spécifier le bloc de routage interdomaine `IPv4` sans classe (CIDR) à partir duquel Kubernetes attribue les adresses IP de service, spécifiez l'option CIDR du service. IPv4

  La spécification de votre propre plage peut aider à éviter les conflits entre les services Kubernetes et les autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

    Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne le spécifiez pas, Kubernetes attribue des adresses IP de service à partir des blocs CIDR ou CIDR. `10.100.0.0/16` `172.20.0.0/16`

- Si vous créez un cluster et que vous souhaitez que le cluster attribue des `IPv6` adresses aux pods et aux services au lieu d'`IPv4`adresses, spécifiez l'option IPFamily.

  Kubernetes attribue des `IPv4` adresses aux pods et aux services, par défaut. Avant de décider d'utiliser la `IPv6` famille, assurez-vous de connaître toutes les considérations et exigences des the section called "IPv6" rubriques the section called "Exigences et considérations requises pour le VPC"the section called "Exigences et considérations requises pour les sous-réseaux",the section called "Exigences de groupe de sécurité", et. Si vous choisissez la `IPv6` famille, vous ne pouvez pas spécifier de plage d'adresses pour que Kubernetes puisse attribuer des adresses de `IPv6` service comme vous le pouvez pour la famille. `IPv4` Kubernetes attribue des adresses de service à partir de la plage d'adresses locale unique (). `fc00::/7`

Créer un cluster - AWS console

1. Ouvrez la console Amazon EKS.
2. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
3. Sous Options de configuration, sélectionnez Configuration personnalisée
   - Pour plus d'informations sur la création rapide d'un cluster avec le mode automatique EKS, consultezthe section called "Console de gestion".
4. Sous Mode automatique EKS, désactivez l'option Utiliser le mode automatique EKS.
   - Pour plus d'informations sur la création d'un cluster en mode automatique EKS avec une configuration personnalisée, consultezthe section called "Créer un cluster automatique".
5. Sur la page Configurer le cluster, renseignez les champs suivants :
   - Nom : nom de votre cluster. Le nom ne peut contenir que des caractères alphanumériques (en distinguant majuscules et minuscules), des traits d'union et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Rôle IAM du cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé pour permettre au plan de contrôle Kubernetes de gérer AWS les ressources en votre nom.
   - Version Kubernetes : version de Kubernetes à utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous avez besoin d'une version antérieure.
   - Type de support : politique de version de Kubernetes que vous souhaitez définir pour votre cluster. Si vous souhaitez que votre cluster fonctionne uniquement sur une version de support standard, vous pouvez choisir le support standard. Si vous souhaitez que votre cluster bénéficie du support étendu à la fin du support standard d'une version, vous pouvez choisir le support étendu. Si vous sélectionnez une version de Kubernetes actuellement en support étendu, vous ne pouvez pas sélectionner le support standard comme option.
   - Chiffrement des secrets : (facultatif) choisissez d'activer le chiffrement des secrets Kubernetes à l'aide d'une clé KMS. Vous pouvez également activer cette option une fois que vous avez créé votre cluster. Avant d'activer cette fonctionnalité, assurez-vous de connaître les informations contenues dansthe section called "Activer le chiffrement secret".
   - Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ".

- Déplacement de zone ARC - (facultatif) Vous pouvez utiliser le contrôleur Route53 Application Recovery pour atténuer les zones de disponibilité altérées. Pour de plus amples informations, veuillez consulter the section called "En savoir plus sur Zonal Shift".

6. Dans la section Accès au cluster de la page de configuration du cluster, entrez les champs suivants :
   - Accès administrateur au cluster Bootstrap : le créateur du cluster est automatiquement un administrateur Kubernetes. Si vous souhaitez désactiver cette option, sélectionnez Interdire l'accès de l'administrateur du cluster.
   - Mode d'authentification du cluster : déterminez comment vous souhaitez accorder aux utilisateurs et aux rôles IAM l'accès à APIs Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Définir le mode d'authentification du cluster".

     Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.
7. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :
   - VPC : choisissez un VPC existant qui répond aux exigences relatives au VPC Amazon EKS pour y créer votre cluster. Avant de choisir un VPC, nous vous recommandons de connaître toutes les exigences et considérations relatives à. the section called "Exigences requises pour le VPC et les sous-réseaux" Vous ne pouvez pas modifier le VPC que vous souhaitez utiliser après la création du cluster. Si aucun VPCs n'est répertorié, vous devez d'abord en créer un. Pour de plus amples informations, veuillez consulter the section called "Création d'un VPC".
   - Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Vous devez en sélectionner au moins deux.

     Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner des sous-réseaux, nous vous recommandons de connaître toutes les exigences et considérations relatives aux VPC et aux sous-réseaux Amazon EKS.

     Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS.

     Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section

called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.

- Choisissez la famille d'adresses IP du cluster : vous pouvez choisir l'une ou l'autre IPv4et IPv6.

  Kubernetes attribue des `IPv4` adresses aux pods et aux services, par défaut. Avant de décider d'utiliser la `IPv6` famille, assurez-vous de connaître toutes les considérations et exigences énoncées dans les exigencesthe section called "Exigences et considérations requises pour les sous-réseaux", the section called "Exigences de groupe de sécurité" les considérations et les rubriques relatives aux VPC. the section called "IPv6" Si vous choisissez la `IPv6` famille, vous ne pouvez pas spécifier de plage d'adresses pour que Kubernetes puisse attribuer des adresses de `IPv6` service comme vous le pouvez pour la famille. `IPv4` Kubernetes attribue des adresses de service à partir de la plage d'adresses locale unique (). `fc00::/7`

- (Facultatif) Choisissez Configurer la plage d'adresses IP du service Kubernetes et spécifiez une gamme de services **`IPv4`**.

  La spécification de votre propre plage peut aider à éviter les conflits entre les services Kubernetes et les autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

  Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne le spécifiez pas, Kubernetes attribue des adresses IP de service à partir des blocs CIDR ou CIDR. `10.100.0.0/16` `172.20.0.0/16`

- Pour Accès au point de terminaison de cluster, sélectionnez une option. Une fois votre cluster créé, vous pouvez modifier cette option. Avant de sélectionner une option autre que par défaut, assurez-vous de vous familiariser avec les options et leurs implications. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

8. (Facultatif) Sur la page Configurer l'observabilité, choisissez quelles options de Métriques et de Journalisation du plan de contrôle doivent être activées. Par défaut, chaque type de journal est désactivé.

- Pour plus d'informations sur l'option Prometheus metrics, consultez. the section called "Étape 1 : activer les métriques Prometheus"
- Pour plus d'informations sur les options de Journalisation du plan de contrôle, consultez the section called "Journaux du plan de contrôle".

Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

9. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster. Certains modules complémentaires sont présélectionnés. Vous pouvez choisir autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez. Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez cliquer sur la numérotation des pages pour afficher des résultats supplémentaires ou rechercher les modules complémentaires AWS Marketplace disponibles en saisissant du texte dans le champ de recherche. Vous pouvez également filtrer par catégorie, fournisseur ou modèle de tarification, puis choisir les modules complémentaires dans les résultats de recherche. Lorsque vous créez un cluster, vous pouvez afficher, sélectionner et installer tout module complémentaire compatible avec EKS Pod Identities, comme indiqué dansthe section called "Identité du pod".

   Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

   Certains modules complémentaires, tels qu'Amazon VPC CNI, CoreDNS et kube-proxy, sont installés par défaut. Si vous désactivez l'un des modules complémentaires par défaut, cela peut affecter votre capacité à exécuter des applications Kubernetes.

10. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez la version que vous voulez installer. Vous pourrez toujours effectuer une mise à jour vers une version ultérieure après la création du cluster.

    Pour les modules complémentaires compatibles avec EKS Pod Identities, vous pouvez utiliser la console pour générer automatiquement le rôle avec le nom, la politique AWS gérée et la politique de confiance préremplis spécifiquement pour le module complémentaire. Vous pouvez réutiliser les rôles existants ou créer de nouveaux rôles pour les modules complémentaires pris en charge. Pour les étapes d'utilisation de la console afin de créer des rôles pour les modules complémentaires compatibles avec EKS Pod Identities, voirthe section called "Créer un module complémentaire (AWS console)". Si un module complémentaire ne prend pas en charge EKS Pod Identity, un message contenant les instructions d'utilisation de l'assistant pour créer les rôles IAM pour les comptes de service (IRSA) s'affiche une fois le cluster créé.

Vous pourrez mettre à jour la configuration de chaque module complémentaire après la création du cluster. Pour plus d'informations sur la configuration des modules complémentaires, consultez the section called "Mettre à jour un module complémentaire". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

11. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez Modifier. Lorsque vous êtes satisfait, choisissez Create. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné.

> **Note**
>
> Vous pouvez recevoir un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

L'approvisionnement de cluster dure plusieurs minutes.

12. Continuez avec the section called "Étape 3 : Mettre à jour kubeconfig"

Créer un cluster - AWS CLI

1. Créez votre cluster à l'aide de la commande suivante. Avant d'exécuter la commande, effectuez les remplacements suivants :
   - *`region-code`* Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
   - Remplacez *`my-cluster`* par un nom pour votre cluster. Le nom ne peut contenir que des caractères alphanumériques (en distinguant majuscules et minuscules), des traits d'union et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - *`1.33`* Remplacez-le par n'importe quel eks/latest/userguide/kubernetes fichier -versions.html [version prise en charge par Amazon EKS, type="documentation "].

- Remplacez *111122223333* par l'ID de votre compte et *myAmazonEKSClusterRole* par le nom de votre rôle IAM de cluster.
- Remplacez les valeurs de `subnetIds` par vos propres valeurs. Vous pouvez également en ajouter d'autres IDs. Vous devez spécifier au moins deux sous-réseaux IDs.

  Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner des sous-réseaux, nous vous recommandons de connaître toutes les exigences et considérations relatives aux VPC et aux sous-réseaux Amazon EKS.
- Si vous ne souhaitez pas spécifier d'ID de groupe de sécurité, `,securityGroupIds=sg-<ExampleID1>` supprimez-le de la commande. Si vous souhaitez spécifier un ou plusieurs groupes de sécurité IDs, remplacez les valeurs de par `securityGroupIds` les vôtres. Vous pouvez également en ajouter d'autres IDs.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-version
 1.33 \
   --role-arn arn:aws: iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=subnet-ExampleID1,subnet-
ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> Vous pouvez recevoir un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

Les paramètres suivants sont des paramètres facultatifs qui, si nécessaire, doivent être ajoutés à la commande précédente. Vous pouvez activer ces options uniquement lorsque vous créez le cluster, et non après.

- Par défaut, EKS installe plusieurs modules complémentaires réseau lors de la création du cluster. Cela inclut Amazon VPC CNI, CoreDNS et kube-proxy.

  Si vous souhaitez désactiver l'installation de ces modules complémentaires réseau par défaut, utilisez le paramètre ci-dessous. Cela peut être utilisé comme alternative CNIs, comme le cilium. Consultez la référence de l'API EKS pour plus d'informations.

  ```
  aws eks create-cluster --bootstrapSelfManagedAddons false
  ```

- Si vous souhaitez spécifier à partir de quel bloc de routage inter-domaines sans classe (CIDR) `IPv4` Kubernetes attribue des adresses IP de service, vous devez le spécifier en ajoutant le `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` à la commande suivante.

  La spécification de votre propre plage peut aider à éviter les conflits entre les services Kubernetes et les autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne le spécifiez pas, Kubernetes attribue des adresses IP de service à partir des blocs CIDR ou CIDR. `10.100.0.0/16` `172.20.0.0/16`

- Si vous créez un cluster et que vous souhaitez que le cluster attribue des `IPv6` adresses aux pods et aux services au lieu d'`IPv4`adresses, ajoutez `--kubernetes-network-config ipFamily=ipv6` la commande suivante.

  Kubernetes attribue des `IPv4` adresses aux pods et aux services, par défaut. Avant de décider d'utiliser la `IPv6` famille, assurez-vous de connaître toutes les considérations et exigences énoncées dans les exigencesthe section called "Exigences et considérations requises pour les

sous-réseaux", the section called "Exigences de groupe de sécurité" les considérations et les rubriques relatives aux VPC. the section called "IPv6" Si vous choisissez la `IPv6` famille, vous ne pouvez pas spécifier de plage d'adresses pour que Kubernetes puisse attribuer des adresses de `IPv6` service comme vous le pouvez pour la famille. `IPv4` Kubernetes attribue des adresses de service à partir de la plage d'adresses locale unique (). `fc00::/7`

2. Le provisionnement du cluster prend quelques minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

   Ne passez pas à l'étape suivante tant que le résultat renvoyé ne l'est pas`ACTIVE`.

3. Continuez avec the section called "Étape 3 : Mettre à jour kubeconfig"

Étape 3 : Mettre à jour kubeconfig

1. Si vous avez créé votre cluster à l'aide de `eksctl`, vous pouvez ignorer cette étape. Cela est dû au fait que `eksctl` a déjà terminé cette étape pour vous. Activez `kubectl` pour communiquer avec votre cluster en ajoutant un nouveau contexte au fichier `kubectl config`. Pour plus d'informations sur la création et la mise à jour du fichier, consultez the section called "Cluster d'accès avec kubectl".

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

   L'exemple qui suit illustre un résultat.

```
Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

2. Confirmez la communication avec votre cluster en exécutant la commande suivante.

```
kubectl get svc
```

   L'exemple qui suit illustre un résultat.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
```

```
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Étape 4 : Configuration du cluster

1. (Recommandé) Pour utiliser certains modules complémentaires Amazon EKS ou pour permettre à des charges de travail Kubernetes individuelles de disposer d'autorisations spécifiques de gestion des AWS identités et des accès (IAM), créez un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Vous n'avez besoin de créer un fournisseur OIDC IAM pour votre cluster qu'une seule fois. Pour en savoir plus sur les modules complémentaires Amazon EKS, consultez the section called "Modules complémentaires Amazon EKS". Pour en savoir plus sur l'attribution des autorisations IAM spécifiques à vos applications, consultez the section called "Identifiants auprès de l'IRSA".
2. (Recommandé) Configurez votre cluster pour le plugin Amazon VPC CNI pour Kubernetes avant de déployer des nœuds Amazon sur EC2 votre cluster. Par défaut, le plugin a été installé avec votre cluster. Lorsque vous ajoutez EC2 des nœuds Amazon à votre cluster, le plugin est automatiquement déployé sur chaque EC2 nœud Amazon que vous ajoutez. Le plug-in vous oblige à associer l'une des politiques IAM suivantes à un rôle IAM. Si votre cluster utilise cette `IPv4` famille, utilisez la politique IAM gérée Amazoneks_CNI_Policy. Si votre cluster utilise cette `IPv6` famille, utilisez une politique IAM que vous créez.

   Le rôle IAM auquel vous attachez la politique peut être le rôle IAM du nœud ou un rôle dédié utilisé uniquement pour le plugin. Nous vous recommandons d'associer la politique à ce rôle. Pour plus d'informations sur la création du rôle, consultez notre article the section called "Configuration pour IRSA" sur le rôle IAM du nœud Amazon EKS.
3. Si vous avez déployé votre cluster à l'aide du AWS Management Console, vous pouvez ignorer cette étape. AWS Management Console Déploie le plug-in Amazon VPC CNI pour les modules complémentaires Kubernetes, CoreDNS et Amazon EKS, par défaut. `kube-proxy`

   Si vous déployez votre cluster à l'aide de l'une `eksctl` ou de l'autre ou de la AWS CLI, le plug-in Amazon VPC CNI pour Kubernetes, CoreDNS et les modules complémentaires autogérés sont déployés. `kube-proxy` Vous pouvez migrer le plug-in Amazon VPC CNI pour Kubernetes, CoreDNS `kube-proxy` et les modules complémentaires autogérés déployés avec votre cluster vers des modules complémentaires Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires Amazon EKS".

4. (Facultatif) Si ce n'est pas déjà fait, vous pouvez activer les métriques Prometheus pour votre cluster. Pour plus d'informations, consultez Création d'un scraper dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.
5. Si vous prévoyez de déployer des charges de travail sur votre cluster qui utilisent des volumes Amazon EBS, vous devez installer Amazon EBS CSI sur votre cluster avant de déployer les charges de travail.

Étapes suivantes

- Le principal IAM qui a créé le cluster est le seul principal à avoir accès au cluster. Accordez des autorisations aux autres principaux IAM afin qu'ils puissent accéder à votre cluster.
- Si le principal IAM qui a créé le cluster ne dispose que des autorisations IAM minimales référencées dans les prérequis, vous pouvez alors ajouter des autorisations Amazon EKS supplémentaires pour ce principal. Pour plus d'informations sur l'octroi d'autorisations Amazon EKS aux principaux IAM, consultez la rubrique the section called “Référence IAM”.
- Si vous souhaitez que le principal IAM qui a créé le cluster, ou tout autre principal, puisse consulter les ressources Kubernetes dans la console Amazon EKS, accordez les autorisations requises aux entités.
- Si vous souhaitez que les nœuds et les principaux IAM accèdent à votre cluster depuis votre VPC, activez le point de terminaison privé pour votre cluster. Le point de terminaison public est activé par défaut. Vous pouvez désactiver le point de terminaison public une fois que vous avez activé le point de terminaison privé, si vous le souhaitez. Pour de plus amples informations, veuillez consulter the section called “Accès au point de terminaison de cluster”.
- Activer le chiffrement des secrets pour votre cluster.
- Configurer la journalisation de votre cluster.
- Ajouter des nœuds à votre cluster.

Préparez-vous aux mises à niveau des versions de Kubernetes et corrigez les erreurs de configuration grâce aux informations du cluster

Amazon EKS Cluster Insights détecte les problèmes et fournit des recommandations pour les résoudre afin de vous aider à gérer votre cluster. Chaque cluster Amazon EKS est soumis à

des contrôles automatiques et récurrents par rapport à une liste d'informations organisée par Amazon EKS. Ces vérifications d'informations sont entièrement gérées par Amazon EKS et proposent des recommandations sur la manière de traiter les éventuels résultats.

Types d'informations sur les clusters

- Informations sur la configuration : identifie les erreurs de configuration dans la configuration de vos nœuds hybrides EKS susceptibles de nuire au fonctionnement de votre cluster ou à vos charges de travail.
- Informations sur les mises à niveau : identifie les problèmes susceptibles d'avoir un impact sur votre capacité à effectuer une mise à niveau vers les nouvelles versions de Kubernetes.

Considérations

- Fréquence : Amazon EKS actualise les informations du cluster toutes les 24 heures. Vous ne pouvez pas actualiser manuellement les informations du cluster. Si vous corrigez un problème de cluster, la mise à jour des informations sur le cluster prendra un certain temps. Pour déterminer si un correctif a réussi, comparez l'heure à laquelle la modification a été déployée à la « date de dernière actualisation » de la vérification des informations.
- Autorisations : Amazon EKS crée automatiquement une entrée d'accès au cluster pour obtenir des informations sur le cluster dans chaque cluster EKS. Cette entrée autorise EKS à consulter les informations relatives à votre cluster. EKS utilise ces informations pour générer des informations. Pour de plus amples informations, veuillez consulter the section called "Amazon EKSCluster InsightsPolicy".

Cas d'utilisation

Les informations sur les clusters d'Amazon EKS fournissent des contrôles automatisés pour vous aider à maintenir l'intégrité, la fiabilité et la configuration optimale de vos clusters Kubernetes. Vous trouverez ci-dessous les principaux cas d'utilisation pour obtenir des informations sur les clusters, notamment la préparation à la mise à niveau et le dépannage de la configuration.

Informations sur les améliorations

Les informations de mise à niveau sont un type spécifique de vérification des informations dans le cadre des informations sur les clusters. Ces vérifications fournissent des informations relatives

à l'état de préparation à la mise à niveau de la version de Kubernetes. Amazon EKS effectue des vérifications des informations de mise à niveau sur chaque cluster EKS.

> **Important**
>
> Amazon EKS a temporairement annulé une fonctionnalité qui vous obligeait à utiliser un `--force` indicateur pour mettre à niveau votre cluster en cas de problèmes d'analyse du cluster. Pour plus d'informations, voir Annulation temporaire de l'application des informations de mise à niveau lors de la mise à jour de la version du cluster sur. GitHub
> Pour plus d'informations sur la mise à jour de votre cluster, consultezthe section called "Étape 3 : Mettre à jour le plan de contrôle du cluster".

Avant de mettre à jour la version Kubernetes de votre cluster, vous pouvez utiliser l'onglet Cluster Insights du tableau de bord d'observabilité de la console Amazon EKS. Si votre cluster a identifié des problèmes, passez-les en revue et apportez les correctifs appropriés. Les problèmes incluent des liens vers la documentation Amazon EKS et Kubernetes. Une fois le problème résolu, attendez que les informations du cluster soient actualisées. Si tous les problèmes ont été résolus, mettez à jour votre cluster.

Amazon EKS fournit des informations relatives à la préparation à la mise à niveau des versions de Kubernetes. Les informations sur les mises à niveau identifient les problèmes potentiels susceptibles d'avoir un impact sur les mises à niveau des clusters Kubernetes. Cela minimise les efforts que les administrateurs consacrent à la préparation des mises à niveau et augmente la fiabilité des applications sur les nouvelles versions de Kubernetes. Les clusters sont automatiquement analysés par Amazon EKS en fonction d'une liste des éventuels problèmes liés à la mise à niveau de la version de Kubernetes. Amazon EKS met fréquemment à jour la liste des vérifications analytiques en fonction de l'examen des modifications apportées à chaque version de Kubernetes.

Les informations de mise à niveau d'Amazon EKS accélèrent le processus de test et de vérification des nouvelles versions. Ils permettent également aux administrateurs de clusters et aux développeurs d'applications de tirer parti des fonctionnalités les plus récentes de Kubernetes en mettant en évidence les problèmes et en proposant des conseils pour y remédier.

Informations sur la configuration

EKS Cluster Insights analyse automatiquement les clusters Amazon EKS dotés de nœuds hybrides afin d'identifier les problèmes de configuration entravant la plane-to-webhook communication

de contrôle Kubernetes, les commandes kubectl telles que exec et logs, et bien plus encore. Les informations de configuration permettent de détecter les problèmes et de fournir des recommandations pour y remédier, accélérant ainsi le délai de configuration de nœuds hybrides entièrement fonctionnels.

Mise en route

Pour consulter la liste des vérifications d'informations effectuées et les éventuels problèmes pertinents identifiés par Amazon EKS, vous pouvez utiliser le AWS Management Console fonctionnement de la AWS CLI et de l'`ListInsights`API Amazon EKS. AWS SDKs Consultez the section called "Afficher les informations sur les clusters" pour démarrer.

Afficher les informations sur les clusters

Amazon EKS fournit deux types d'informations : des informations sur la configuration et des informations sur les mises à niveau. Les informations de configuration identifient les erreurs de configuration dans la configuration de vos nœuds hybrides EKS susceptibles de nuire au fonctionnement de votre cluster ou à vos charges de travail. Les informations sur les mises à niveau identifient les problèmes susceptibles d'avoir un impact sur votre capacité à effectuer une mise à niveau vers les nouvelles versions de Kubernetes.

Pour consulter la liste des vérifications d'informations effectuées et les éventuels problèmes pertinents identifiés par Amazon EKS, vous pouvez appeler le fonctionnement de look in the AWS Management Console, de la AWS CLI et de l'`ListInsights`API Amazon EKS. AWS SDKs

Afficher les informations relatives à la configuration (console)

1. Ouvrez la console Amazon EKS.
2. Dans la liste de clusters, choisissez le nom du cluster Amazon EKS pour lequel vous souhaitez obtenir des informations.
3. Choisissez Monitor cluster.
4. Choisissez l'onglet État du cluster.
5. Dans le tableau des informations sur la configuration, vous verrez les colonnes suivantes :
   - Nom : vérification effectuée par Amazon EKS par rapport au cluster.
   - État des informations : une information dont le statut est égal à `Error` indique qu'une mauvaise configuration est susceptible d'avoir un impact sur les fonctionnalités du cluster. Un aperçu

dont le statut est égal à `Warning` signifie que la configuration ne correspond pas à l'approche documentée, mais que la fonctionnalité du cluster peut fonctionner si vous l'avez configurée intentionnellement. Un aperçu avec un statut de `Passing` indique qu'Amazon EKS n'a détecté aucun problème associé à cette vérification d'aperçu dans votre cluster.

- Version : version applicable.
- Heure de dernière actualisation : heure à laquelle le statut de l'aperçu a été actualisé pour la dernière fois pour ce cluster.
- Description : informations provenant de la vérification des informations, qui incluent l'alerte et les mesures correctives recommandées.

Afficher les informations relatives aux mises à niveau (Console)

1. Ouvrez la console Amazon EKS.
2. Dans la liste de clusters, choisissez le nom du cluster Amazon EKS pour lequel vous souhaitez obtenir des informations.
3. Choisissez Monitor cluster.
4. Choisissez l'onglet Upgrade Insights.
5. Dans le tableau des informations sur les mises à niveau, vous verrez les colonnes suivantes :
   - Nom : vérification effectuée par Amazon EKS par rapport au cluster.
   - État de l'aperçu : un aperçu dont le statut est « Erreur » signifie généralement que la version de Kubernetes affectée est N+1 de la version actuelle du cluster, tandis qu'un statut « Avertissement » signifie que l'aperçu s'applique à une future version de Kubernetes N+2 ou plus. Une information dont le statut est « Réussi » signifie qu'Amazon EKS n'a détecté aucun problème associé à cette vérification d'informations dans votre cluster. Une information dont le statut est « Inconnu » signifie qu'Amazon EKS n'est pas en mesure de déterminer si votre cluster est concerné par cette vérification des informations.
   - Version : version de Kubernetes pour laquelle l'analyse a vérifié l'absence de problèmes éventuels.
   - Heure de dernière actualisation : heure à laquelle le statut de l'aperçu a été actualisé pour la dernière fois pour ce cluster.
   - Heure de la dernière transition : heure à laquelle le statut de cet aperçu a été modifié pour la dernière fois.
   - Description : informations provenant de la vérification des informations, qui incluent l'alerte et les mesures correctives recommandées.

Afficher les informations sur le cluster (AWS CLI)

1. Déterminez le cluster que vous souhaitez vérifier pour obtenir des informations. La commande suivante répertorie les information d'un cluster spécifique. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :
   - *region-code*Remplacez-le par le code de votre AWS région.
   - Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks list-insights --region region-code --cluster-name my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   {
   "insights":
       [
           {
               "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
               "name": "Deprecated APIs removed in Kubernetes vX.XX",
               "category": "UPGRADE_READINESS",
               "kubernetesVersion": "X.XX",
               "lastRefreshTime": 1734557315.000,
               "lastTransitionTime": 1734557309.000,
               "description": "Checks for usage of deprecated APIs that are scheduled
    for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
    updated APIs supported by vX.XX could cause application impact.",
               "insightStatus":
               {
                   "status": "PASSING",
                   "reason": "No deprecated API usage detected within the last 30
    days.",
               },
           },
           {
               "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
               "name": "Kubelet version skew",
               "category": "UPGRADE_READINESS",
               "kubernetesVersion": "X.XX",
               "lastRefreshTime": 1734557309.000,
               "lastTransitionTime": 1734557309.000,
   ```

```
            "description": "Checks for kubelet versions of worker nodes in the
cluster to see if upgrade would cause non compliance with supported Kubernetes
kubelet version skew policy.",
            "insightStatus":
            {
                "status": "UNKNOWN",
                "reason": "Unable to determine status of node kubelet versions.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "Cluster health issues",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for any cluster health issues that prevent
successful upgrade to the next Kubernetes version on EKS.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No cluster health issues detected.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "EKS add-on version compatibility",
```

```
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of installed EKS add-ons to ensure they
are compatible with the next version of Kubernetes. ",
            "insightStatus": { "status": "PASSING", "reason": "All installed EKS
add-on versions are compatible with next Kubernetes version."},
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "name": "kube-proxy version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of kube-proxy in cluster to see if
upgrade would cause non compliance with supported Kubernetes kube-proxy version
skew policy.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "kube-proxy versions match the cluster control plane
version.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEddddd",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
    ],
```

```
"nextToken": null,
}
```

2. Pour obtenir une description des informations, exécutez la commande suivante. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :
   - *region-code*Remplacez-le par le code de votre AWS région.
   - *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222*Remplacez-le par un ID d'aperçu extrait de la liste des informations du cluster.
   - Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222 --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.

```
{
  "insight":
    {
      "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "name": "Kubelet version skew",
      "category": "UPGRADE_READINESS",
      "kubernetesVersion": "1.27",
      "lastRefreshTime": 1734557309.000,
      "lastTransitionTime": 1734557309.000,
      "description": "Checks for kubelet versions of worker nodes in the cluster
 to see if upgrade would cause non compliance with supported Kubernetes kubelet
 version skew policy.",
      "insightStatus":
        {
          "status": "UNKNOWN",
          "reason": "Unable to determine status of node kubelet versions.",
        },
      "recommendation": "Upgrade your worker nodes to match the Kubernetes version
 of your cluster control plane.",
      "additionalInfo":
        {
          "Kubelet version skew policy": "https://kubernetes.io/releases/version-
skew-policy/#kubelet",
          "Updating a managed node group": "https://docs.aws.amazon.com/eks/latest/
userguide/update-managed-node-group.html",
        },
```

```
      "resources": [],
      "categorySpecificSummary":
        { "deprecationDetails": [], "addonCompatibilityDetails": [] },
    },
}
```

Mettre à jour le cluster existant vers la nouvelle version de Kubernetes

Dès qu'une nouvelle version Kubernetes est disponible dans Amazon EKS, vous pouvez mettre à jour votre cluster Amazon EKS.

> **Important**
>
> Une fois que vous avez mis à niveau un cluster, vous ne pouvez pas le rétrograder vers une version précédente. Avant de passer à une nouvelle version de Kubernetes, nous vous recommandons de consulter les informations contenues dans le eks/latest/userguide/kubernetes fichier -versions.html [Comprendre le cycle de vie des versions de Kubernetes sur EKS, type="documentation "] et les étapes de mise à jour décrites dans cette rubrique.

Les nouvelles versions de Kubernetes introduisent parfois des modifications importantes. Nous vous recommandons donc de comparer la version actuelle de vos applications à la toute nouvelle version Kubernetes avant de procéder à la mise à jour sur vos clusters de production. Pour ce faire, créez un flux d'intégration continue afin de tester le comportement de votre application avant de passer à une nouvelle version Kubernetes.

Dans le processus de mise à jour, Amazon EKS lance de nouveaux nœuds de serveur d'API avec la version de Kubernetes mise à jour pour remplacer les versions existantes. Amazon EKS effectue des vérifications standard de l'état de l'infrastructure et de l'état de préparation du trafic réseau sur ces nouveaux nœuds afin de vérifier qu'ils fonctionnent comme prévu. Cependant, une fois que vous avez commencé la mise à niveau du cluster, vous ne pouvez ni la suspendre ni l'arrêter. Si l'une de ces vérifications échoue, Amazon EKS annule le déploiement d'infrastructure, et votre cluster reste dans la version Kubernetes précédente. Les applications en cours d'exécution ne sont pas affectées et votre cluster n'est jamais laissé dans un état non déterministe ou irrécupérable. Amazon EKS sauvegarde régulièrement tous les clusters gérés, et des mécanismes existent pour récupérer

des clusters si nécessaire. Nous évaluons et améliorons constamment nos processus de gestion de l'infrastructure Kubernetes.

Pour mettre à jour le cluster, Amazon EKS requiert jusqu'à cinq adresses IP disponibles correspondant aux sous-réseaux que vous avez spécifiés lors de la création de votre cluster. Amazon EKS crée de nouvelles interfaces réseau élastiques de cluster (interfaces réseau) dans l'un des sous-réseaux que vous avez spécifiés. Les interfaces réseau peuvent être créées dans des sous-réseaux différents de ceux dans lesquels se trouvent vos interfaces réseau existantes. Assurez-vous donc que les règles de votre groupe de sécurité autorisent la communication de cluster requise pour tous les sous-réseaux que vous avez spécifiés lors de la création de votre cluster. Si l'un des sous-réseaux que vous avez spécifiés lors de la création du cluster n'existe pas, s'il n'a pas suffisamment d'adresses IP disponibles ou s'il n'existe pas de règles de groupe de sécurité autorisant les communications nécessaires au cluster, la mise à jour peut échouer.

Pour garantir que le point de terminaison du serveur d'API de votre cluster est toujours accessible, Amazon EKS fournit un plan de contrôle Kubernetes à haute disponibilité et effectue des mises à jour continues des instances du serveur d'API pendant les opérations de mise à jour. Afin de prendre en compte les modifications des adresses IP des instances de serveur d'API prenant en charge le point de terminaison de votre serveur d'API Kubernetes, vous devez vous assurer que les clients de votre serveur d'API gèrent efficacement les reconnexions. Les versions récentes `kubectl` et les bibliothèques clientes de Kubernetes officiellement prises en charge exécutent ce processus de reconnexion de manière transparente.

> **Note**
>
> Pour en savoir plus sur le contenu d'une mise à jour de cluster, consultez les meilleures pratiques pour les mises à niveau de clusters dans le guide des meilleures pratiques d'EKS. Cette ressource vous aide à planifier une mise à niveau et à comprendre la stratégie de mise à niveau d'un cluster.

Considérations relatives au mode automatique d'Amazon EKS

- La capacité de calcul du mode automatique d'Amazon EKS contrôle la version Kubernetes des nœuds. Après la mise à niveau du plan de contrôle, le mode automatique d'EKS commence à mettre à jour progressivement les nœuds gérés. Le mode automatique EKS respecte les budgets d'interruption des pods.

- Il n'est pas nécessaire de mettre à niveau manuellement les fonctionnalités d'Amazon EKS Auto Mode, notamment les fonctionnalités de mise à l'échelle automatique du calcul, de stockage par blocs et d'équilibrage de charge.

Récapitulatif

Le résumé général du processus de mise à niveau du cluster Amazon EKS est le suivant :

1. Assurez-vous que votre cluster est dans un état permettant une mise à niveau. Cela inclut la vérification des Kubernetes APIs utilisés par les ressources déployées dans le cluster, afin de s'assurer que le cluster ne présente aucun problème de santé. Vous devez utiliser les informations de mise à niveau d'Amazon EKS pour évaluer l'état de préparation de votre cluster à la mise à niveau.
2. Mettez à niveau le plan de contrôle vers la version mineure suivante (par exemple, de la version 1.32 à la version 1.33).
3. Mettez à niveau les nœuds du plan de données pour qu'ils correspondent à ceux du plan de contrôle.
4. Mettez à niveau toutes les applications supplémentaires qui s'exécutent sur le cluster (par exemple,`cluster-autoscaler`).
5. Mettez à niveau les modules complémentaires fournis par Amazon EKS, tels que ceux inclus par défaut :
   - Version recommandée d'Amazon VPC CNI
   - Version recommandée de CoreDNS
   - `kube-proxy`version recommandée
6. Mettez à niveau tous les clients qui communiquent avec le cluster (par exemple,`kubectl`).

Étape 1 : Préparation de la mise à niveau

Comparez la version Kubernetes de votre plan de contrôle de cluster à celle de vos nœuds.

- Obtenez la version Kubernetes de votre plan de contrôle de cluster.

```
kubectl version
```

- Obtenez la version Kubernetes de vos nœuds. Cette commande renvoie tous les nœuds Amazon EC2, Fargate et hybrides autogérés et gérés. Chaque Fargate Pod est répertorié comme son propre nœud.

```
kubectl get nodes
```

Avant de mettre à jour votre plan de contrôle vers une nouvelle version de Kubernetes, assurez-vous que la version mineure de Kubernetes des nœuds gérés et des nœuds Fargate de votre cluster est identique à la version de votre plan de contrôle. Par exemple, si votre plan de contrôle exécute la version `1.29` et que l'un de vos nœuds exécute la version`1.28`, vous devez mettre à jour vos nœuds vers la version 1.30 `1.29` avant de mettre à jour votre plan de contrôle vers la version 1.30. Nous vous recommandons également de mettre à jour vos nœuds autogérés et vos nœuds hybrides vers la même version que votre plan de contrôle avant de mettre à jour le plan de contrôle. Pour plus d'informations, consultez the section called "Mettre à jour", the section called "Méthodes de mise à jour" et the section called "Mettre à niveau les nœuds hybrides". Si vous avez des nœuds Fargate dont la version mineure est inférieure à la version du plan de contrôle, supprimez d'abord le pod représenté par le nœud. Mettez ensuite à jour votre plan de contrôle. Tous les pods restants seront mis à jour vers la nouvelle version une fois que vous les aurez redéployés.

Étape 2 : examiner les considérations relatives à la mise à niveau

Amazon EKS Cluster Insights analyse automatiquement les clusters par rapport à une liste de mises à niveau potentielles des versions de Kubernetes ayant un impact sur des problèmes tels que l'utilisation obsolète de l'API Kubernetes. Amazon EKS met régulièrement à jour la liste des vérifications d'informations à effectuer en fonction des évaluations des modifications apportées au projet Kubernetes. Amazon EKS met également à jour la liste de contrôle des informations au fur et à mesure que des modifications sont introduites dans le service Amazon EKS ainsi que des nouvelles versions. Pour de plus amples informations, veuillez consulter the section called "Informations sur le cluster".

Consultez le guide de migration d'API obsolète dans la documentation de Kubernetes.

Consultez les informations de mise à niveau

Utilisez les informations de mise à niveau d'Amazon EKS pour identifier les problèmes. Pour de plus amples informations, veuillez consulter the section called "Afficher les informations relatives aux mises à niveau (Console)".

Considérations détaillées

- Dans la mesure où Amazon EKS exécute un plan de contrôle hautement disponible, vous devez mettre à jour une seule version mineure à la fois. Pour plus d'informations sur cette exigence, consultez Kubernetes Version and Version Skew Support Policy (Politique de prise en charge des versions et versions asymétriques de Kubernetes). Supposez que la version actuelle de votre cluster soit la version `1.28` et que vous souhaitiez la mettre à jour vers la version `1.30`. Vous devez d'abord mettre à jour la version `1.28` de votre cluster vers la version `1.29`, puis mettre à jour la version `1.29` de votre cluster vers la version `1.30`.
- Vérifiez l'écart de version entre Kubernetes `kube-apiserver` et celui de vos nœuds. `kubelet`
  - À partir de la version Kubernetes`1.28`, il `kubelet` peut y avoir jusqu'à trois versions mineures antérieures à. `kube-apiserver` Voir la politique d'asymétrie des versions en amont de Kubernetes.
  - Si vos nœuds gérés et Fargate utilisent la `kubelet` version Kubernetes ou une `1.25` version plus récente, vous pouvez mettre à jour votre cluster jusqu'à trois versions à l'avance sans mettre à jour la version. `kubelet` Par exemple, si la version du `kubelet` est `1.25`, vous pouvez mettre à jour la version de votre cluster Amazon EKS de `1.25` vers `1.26` ou vers `1.27` et vers `1.28` alors que le `kubelet` reste sur la version `1.25`.
- Avant de commencer une mise à jour, assurez-vous que la `kubelet` version de Kubernetes de vos nœuds est la même que celle de votre plan de contrôle.
- Si votre cluster est configuré avec une version du plugin Amazon VPC CNI pour Kubernetes antérieure à`1.8.0`, nous vous recommandons de mettre à jour le plug-in avec la dernière version avant de mettre à jour votre cluster. Pour mettre à jour le plugin, consultez the section called "CNI Amazon VPC".

Étape 3 : Mettre à jour le plan de contrôle du cluster

> **Important**
>
> Amazon EKS a temporairement annulé une fonctionnalité qui vous obligeait à utiliser un `--force` indicateur pour mettre à niveau votre cluster en cas de problèmes d'analyse du cluster. Pour plus d'informations, voir Annulation temporaire de l'application des informations de mise à niveau lors de la mise à jour de la version du cluster sur. GitHub

Amazon EKS actualise un aperçu du cluster 24 heures après la « dernière actualisation ». Vous pouvez comparer l'heure à laquelle vous avez résolu un problème à la « date de dernière actualisation » de l'aperçu du cluster.
En outre, la mise à jour du statut d'aperçu peut prendre jusqu'à 30 jours après la résolution d'une utilisation obsolète de l'API. Upgrade Insights recherche toujours les utilisations déconseillées des API sur une période continue de 30 jours.

Vous pouvez soumettre la demande de mise à niveau de la version de votre plan de contrôle EKS en utilisant :

- eksctl
- la AWS console
- la AWS CLI

Mettre à jour le cluster - eksctl

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation et de mise à jour de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

Mettez à jour la version Kubernetes de votre plan de contrôle Amazon EKS. Remplacez `<cluster-name>` par le nom de votre cluster. `<version-number>`Remplacez-le par le numéro de version pris en charge par Amazon EKS vers lequel vous souhaitez mettre à jour votre cluster. Pour obtenir la liste des numéros de version pris en charge, consultez le eks/latest/userguide/kubernetes fichier -versions.html [Versions prises en charge par Amazon EKS, type="documentation "].

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

Cette mise à jour peut prendre plusieurs minutes.

Passez au the section called “Étape 4 : Mettre à jour les composants du cluster”.

Cluster de mise à jour - AWS console

1. Ouvrez la console Amazon EKS.
2. Choisissez Mettre à niveau maintenant pour le cluster que vous souhaitez mettre à niveau.
3. Sélectionnez la version vers laquelle mettre à jour votre cluster et choisissez Mettre à niveau.
4. Cette mise à jour peut prendre plusieurs minutes. Passez au the section called "Étape 4 : Mettre à jour les composants du cluster".

Mettre à jour le cluster - AWS CLI

1. Vérifiez que la AWS CLI est installée et que vous êtes connecté. Pour plus d'informations, voir Installation ou mise à jour vers la dernière version de la AWS CLI.
2. Mettez à jour votre cluster Amazon EKS à l'aide de la commande AWS CLI suivante. Remplacez `<cluster-name>` et `<region-code>` du cluster que vous souhaitez mettre à niveau. `<version-number>`Remplacez-le par le numéro de version pris en charge par Amazon EKS vers lequel vous souhaitez mettre à jour votre cluster. Pour obtenir la liste des numéros de version pris en charge, consultez le eks/latest/userguide/kubernetes fichier -versions.html [Versions prises en charge par Amazon EKS, type="documentation "].

```
aws eks update-cluster-version --name <cluster-name> \
  --kubernetes-version <verion-number> --region <region-code>
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "<update-id>",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "<version-number>"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
```

```
        ],
[...]
        "errors": []
    }
```

3. Cette mise à jour peut prendre plusieurs minutes. Contrôlez l'état de mise à jour de votre cluster avec la commande suivante. En plus d'utiliser le même `<cluster-name>` et`<region-code>`, utilisez `<update-id>` celui renvoyé par la commande précédente.

```
aws eks describe-update --name <cluster-name> \
   --region <region-code> --update-id <update-id>
```

   Lorsqu'un état `Successful` s'affiche, la mise à jour est terminée.

4. Passez au the section called "Étape 4 : Mettre à jour les composants du cluster".

Étape 4 : Mettre à jour les composants du cluster

1. Une fois la mise à jour de votre cluster terminée, mettez à jour vos nœuds vers la même version Kubernetes que celle de votre cluster mis à jour. Pour plus d'informations, consultez the section called "Méthodes de mise à jour", the section called "Mettre à jour" et the section called "Mettre à niveau les nœuds hybrides". Tous les nouveaux pods lancés sur Fargate ont `kubelet` une version qui correspond à la version de votre cluster. Les Fargate Pods existants ne sont pas modifiés.
2. (Facultatif) Si vous avez déployé le composant Kubernetes Cluster Autoscaler sur votre cluster avant de mettre à jour le cluster, mettez ce composant à jour vers la version la plus récente correspondant à la version majeure et mineure de Kubernetes vers laquelle vous avez effectué la mise à jour.
   a. Ouvrez la page des versions de Cluster Autoscaler dans un navigateur Web et recherchez la dernière version de Cluster Autoscaler qui correspond aux versions majeure et mineure de Kubernetes de votre cluster. Par exemple, si la version Kubernetes de votre cluster est, `1.30` recherchez la dernière version de Cluster Autoscaler commençant par. `1.30` Enregistrez le numéro de version sémantique (`1.30.n`, par exemple) de cette version à utiliser à l'étape suivante.
   b. Identifiez l'image Cluster Autoscaler sur la version que vous avez enregistrée à l'étape précédente avec la commande suivante. Si nécessaire, remplacez `X.XX.X` par votre propre valeur.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
```

3. (Clusters dotés de nœuds GPU uniquement) Si votre cluster possède des groupes de nœuds compatibles avec le GPU (par exemple,`p3.2xlarge`), vous devez mettre à jour le plug-in d'appareil NVIDIA pour Kubernetes DaemonSet sur votre cluster. `<vX.X.X>`Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/
deployments/static/nvidia-device-plugin.yml
```

4. Mettez à jour le plug-in Amazon VPC CNI pour Kubernetes, CoreDNS et les modules complémentaires. `kube-proxy` Nous vous recommandons de mettre à jour les modules complémentaires avec les versions minimales répertoriées dans les jetons de compte de service.
   - Si vous utilisez des modules complémentaires Amazon EKS, dans la console Amazon EKS, sélectionnez Clusters, puis le nom du cluster que vous avez mis à jour dans le volet gauche. Les notifications s'affichent dans la console. Elles vous informent qu'une nouvelle version est disponible pour chaque module complémentaire disposant d'une mise à jour disponible. Pour mettre à jour un module complémentaire, sélectionnez l'onglet Add-ons (Modules complémentaires). Dans l'une des zones d'un module complémentaire dont une mise à jour est disponible, sélectionnez Update now (Mettre à jour maintenant), sélectionnez une version disponible, puis cliquez sur Update (Mise à jour).
   - Vous pouvez également utiliser la AWS CLI ou mettre `eksctl` à jour des modules complémentaires. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour un module complémentaire".
5. Si nécessaire, mettez à jour votre version de `kubectl`. Vous devez utiliser une version `kubectl` qui se situe à une différence de version mineure près du plan de contrôle de votre cluster Amazon EKS.

Rétrograder la version Kubernetes pour un cluster Amazon EKS

Vous ne pouvez pas rétrograder les Kubernetes d'un cluster Amazon EKS. Créez plutôt un nouveau cluster sur une version précédente d'Amazon EKS et migrez les charges de travail.

Supprimer un cluster

Lorsque vous avez terminé d'utiliser un cluster Amazon EKS, vous devez supprimer les ressources qui y sont associées afin de ne pas encourir de frais inutiles.

Vous pouvez supprimer un cluster à l'aide `eksctl` de la AWS Management Console CLI ou de la AWS CLI.

Considérations

- Si vous recevez une erreur en raison de la suppression du créateur du cluster, reportez-vous à cet article pour la résoudre.
- Les ressources Amazon Managed Service for Prometheus ne font pas partie du cycle de vie du cluster et doivent être gérées indépendamment du cluster. Lorsque vous supprimez votre cluster, assurez-vous de supprimer également tous les scrapers applicables afin de mettre fin aux coûts applicables. Pour plus d'informations, consultez la section Rechercher et supprimer des scrapers dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.
- Pour supprimer un cluster connecté, reportez-vous à la section the section called "Annuler l'enregistrement d'un cluster"

Considérations relatives au mode automatique EKS

- Tous les nœuds du mode automatique EKS seront supprimés, y compris les instances EC2 gérées
- Tous les équilibreurs de charge seront supprimés

Pour de plus amples informations, veuillez consulter the section called "Désactiver le mode automatique EKS".

Supprimer le cluster (eksctl)

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

1. Répertoriez tous les services qui s'exécutent dans votre cluster.

```
kubectl get svc --all-namespaces
```

   a. Supprimez tous les services qui ont une valeur `EXTERNAL-IP` associée. Ces services ont un équilibreur de charge Elastic Load Balancing en avant-plan, et vous devez les supprimer dans Kubernetes pour que l'équilibreur de charge et les ressources associées soient correctement libérés. Remplacez *`service-name`* par le nom de chaque service répertorié comme décrit.

```
kubectl delete svc service-name
```

2. Supprimez le cluster et les nœuds qui y sont associés avec la commande suivante, en remplaçant *`prod`* par le nom de votre cluster.

```
eksctl delete cluster --name prod
```

   Sortie :

```
[#]  using region region-code
[#]  deleting EKS cluster "prod"
[#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
[#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
[#]  will delete stack "eksctl-prod-cluster"
[#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster. If in
 doubt, check CloudFormation console
```

Supprimer le cluster (AWS console)

1. Répertoriez tous les services qui s'exécutent dans votre cluster.

```
kubectl get svc --all-namespaces
```

2. Supprimez tous les services qui ont une valeur `EXTERNAL-IP` associée. Ces services ont un équilibreur de charge Elastic Load Balancing en avant-plan, et vous devez les supprimer dans Kubernetes pour que l'équilibreur de charge et les ressources associées soient correctement libérés. Remplacez *`service-name`* par le nom de chaque service répertorié comme décrit.

```
kubectl delete svc service-name
```

3. Supprimez tous les groupes de nœuds et les profils Fargate.
   a. Ouvrez la console Amazon EKS.
   b. Dans le panneau de navigation de gauche, sélectionnez Clusters Amazon EKS, puis dans la liste à onglets des clusters, sélectionnez le nom du cluster que vous voulez supprimer.
   c. Sélectionnez l'onglet Calcul et choisissez un groupe de nœuds à supprimer. Sélectionnez Supprimer, saisissez le nom du groupe de nœuds, puis sélectionnez Supprimer. Supprimez tous les groupes de nœuds du cluster.

      > **Note**
      > Tous les groupes de nœuds répertoriés sont des groupes de nœuds gérés.

   d. Sélectionnez un profil Fargate à supprimer, choisissez Supprimer, saisissez le nom du profil, puis sélectionnez Supprimer. Supprimez tous les profils Fargate dans le cluster.
4. Supprimez toutes les piles de nœuds AWS CloudFormation autogérées.
   a. Ouvrez la AWS CloudFormation console.
   b. Sélectionnez la pile du nœud à supprimer, puis choisissez Supprimer.
   c. Dans la boîte de dialogue de confirmation Delete stack (Supprimer la pile), choisissez Delete stack (Supprimer la pile). Supprimez toutes les piles de nœuds autogérées dans le cluster.
5. Supprimez le cluster.
   a. Ouvrez la console Amazon EKS.
   b. Sélectionnez le cluster à supprimer, puis choisissez Supprimer.
   c. Dans l'écran de confirmation de suppression du cluster, choisissez Delete (Supprimer).
6. (Facultatif) Supprimez la pile VPC. AWS CloudFormation
   a. Ouvrez la AWS CloudFormation console.
   b. Sélectionnez la pile VPC à supprimer, puis choisissez Delete (Supprimer).
   c. Dans la boîte de dialogue de confirmation Delete stack (Supprimer la pile), choisissez Delete stack (Supprimer la pile).

Supprimer le cluster (AWS CLI)

1. Répertoriez tous les services qui s'exécutent dans votre cluster.

```
kubectl get svc --all-namespaces
```

2. Supprimez tous les services qui ont une valeur `EXTERNAL-IP` associée. Ces services ont un équilibreur de charge Elastic Load Balancing en avant-plan, et vous devez les supprimer dans Kubernetes pour que l'équilibreur de charge et les ressources associées soient correctement libérés. Remplacez *`service-name`* par le nom de chaque service répertorié comme décrit.

```
kubectl delete svc service-name
```

3. Supprimez tous les groupes de nœuds et les profils Fargate.
   a. Répertoriez les groupes de nœuds de votre cluster à l'aide de la commande suivante.

   ```
   aws eks list-nodegroups --cluster-name my-cluster
   ```

   > **Note**
   > Tous les groupes de nœuds répertoriés sont des groupes de nœuds gérés.

   b. Supprimez chaque groupe de nœuds à l'aide de la commande suivante. Supprimez tous les groupes de nœuds du cluster.

   ```
   aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
   ```

   c. Répétez les profils Fargate dans votre cluster à l'aide de la commande suivante.

   ```
   aws eks list-fargate-profiles --cluster-name my-cluster
   ```

   d. Supprimez chaque profil Fargate avec la commande suivante. Supprimez tous les profils Fargate dans le cluster.

   ```
   aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
   cluster-name my-cluster
   ```

4. Supprimez toutes les piles de nœuds AWS CloudFormation autogérées.
   a. Répertoriez vos AWS CloudFormation piles disponibles à l'aide de la commande suivante. Identifiez le nom du modèle du nœud dans la sortie obtenue.

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

b. Supprimez chaque pile de nœud avec la commande suivante, en remplaçant *node-stack* par le nom de votre pile de nœuds. Supprimez toutes les piles de nœuds autogérées dans le cluster.

```
aws cloudformation delete-stack --stack-name node-stack
```

5. Supprimez le cluster avec la commande suivante, en remplaçant *my-cluster* par le nom de votre cluster.

```
aws eks delete-cluster --name my-cluster
```

6. (Facultatif) Supprimez la pile VPC. AWS CloudFormation

a. Répertoriez vos AWS CloudFormation piles disponibles à l'aide de la commande suivante. Recherchez le nom du modèle VPC dans la sortie obtenue.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. Supprimez la pile VPC avec la commande suivante, en remplaçant *my-vpc-stack* par le nom de votre pile VPC.

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

Protégez les clusters EKS contre les suppressions accidentelles

La suppression accidentelle d'un cluster EKS peut affecter le fonctionnement du cluster Kubernetes.

Vous pouvez désormais protéger les clusters EKS contre toute suppression accidentelle. Si vous activez la protection contre la suppression sur un cluster, vous devez d'abord désactiver la protection contre la suppression avant de pouvoir supprimer le cluster.

Le but de la protection contre les suppressions est de prévenir les accidents. Vous devez limiter avec soin les personnes autorisées à supprimer des clusters.

Si vous essayez de supprimer un cluster actif dont la protection contre la suppression est activée, vous recevrez un`InvalidRequestException`.

> **Important**
>
> Si vous activez la protection contre la suppression sur un cluster, vous devez disposer des autorisations IAM UpdateClusterConfig et des autorisations DeleteCluster IAM pour d'abord supprimer la protection contre la suppression, puis supprimer le cluster.

> **Note**
>
> Si l'état du cluster est en cours de création, d'échec ou de suppression, vous pouvez supprimer le cluster même si la protection contre la suppression est activée.

Pour activer la protection contre les suppressions pour un cluster existant

Vous ne pouvez l'exécuter que sur un cluster dont l'état est actif.

```
aws eks update-cluster-config --deletion-protection
```

Pour désactiver la protection contre les suppressions pour un cluster existant

```
aws eks update-cluster-config --no-deletion-protection
```

Point de terminaison du serveur API du cluster

Cette rubrique vous aide à activer l'accès privé pour le point de terminaison du serveur d'API Kubernetes de votre cluster Amazon EKS et à limiter, voire à désactiver complètement, l'accès public depuis Internet.

Lorsque vous créez un cluster, Amazon EKS crée un point de terminaison pour le serveur d'API Kubernetes géré que vous utilisez pour communiquer avec votre cluster (à l'aide d'outils de gestion Kubernetes comme `kubectl`). Par défaut, ce point de terminaison du serveur d'API est public sur Internet, et l'accès au serveur d'API est sécurisé à l'aide d'une combinaison d' AWS Identity and Access Management (IAM) et de contrôle d'accès basé sur le rôle (RBAC) Kubernetes natif. Ce point de terminaison est connu sous le nom de point de terminaison public du cluster. Il existe également un point de terminaison privé du cluster. Pour plus d'informations sur le point de terminaison privé du cluster, consultez la section suivantethe section called “Point de terminaison privé du cluster”.

IPv6format d'extrémité du cluster

EKS crée un point de terminaison à double pile unique au format suivant pour les nouveaux IPv6 clusters créés après octobre 2024. Un IPv6 cluster est un cluster que vous sélectionnez IPv6 dans le paramètre IP family (ipFamily) du cluster.

Example

AWS

Point de terminaison public/privé du cluster EKS : eks-cluster.*region*.api.aws

AWS GovCloud (US)

Point de terminaison public/privé du cluster EKS : eks-cluster.*region*.api.aws

Amazon Web Services in China

Point de terminaison public/privé du cluster EKS : eks-cluster.*region*.api.amazonwebservices.com.cn

> **Note**
>
> Le point de terminaison du cluster à double pile a été introduit en octobre 2024. Pour plus d'informations sur IPv6 les clusters, consultezthe section called "IPv6". Les clusters créés avant octobre 2024 utilisent plutôt le format de point de terminaison suivant.

IPv4format d'extrémité du cluster

EKS crée un point de terminaison unique au format suivant pour chaque cluster sélectionné IPv4 dans le paramètre de famille d'adresses IP (IPFamily) du cluster :

Example

AWS

Point de terminaison public/privé du cluster EKS eks-cluster.*region*.eks.amazonaws.com

AWS GovCloud (US)

Point de terminaison public/privé du cluster EKS eks-cluster.*region*.eks.amazonaws.com

Amazon Web Services in China

Point de terminaison public/privé du cluster EKS `eks-cluster.region.amazonwebservices.com.cn`

> **Note**
>
> Avant octobre 2024, les `IPv6` clusters utilisaient également ce format de point de terminaison. Pour ces clusters, seules les `IPv4` adresses du point de terminaison public et du point de terminaison privé sont résolues à partir de ce point de terminaison.

Point de terminaison privé du cluster

Vous pouvez activer l'accès privé au serveur d'API Kubernetes pour que toutes les communications entre vos nœuds et le serveur d'API restent au sein de votre VPC. Vous pouvez limiter les adresses IP qui peuvent accéder à votre serveur API à partir d'Internet, ou désactiver complètement l'accès Internet au serveur d'API.

> **Note**
>
> Comme ce point de terminaison est destiné au serveur d'API Kubernetes et n'est pas un point de AWS PrivateLink terminaison traditionnel pour communiquer avec une AWS API, il n'apparaît pas en tant que point de terminaison dans la console Amazon VPC.

Lorsque vous activez l'accès privé aux terminaux pour votre cluster, Amazon EKS crée une zone hébergée privée Route 53 en votre nom et l'associe au VPC de votre cluster. Cette zone hébergée privée est gérée par Amazon EKS et n'apparaît pas dans les ressources Route 53 de votre compte. Pour que la zone hébergée privée achemine correctement le trafic vers votre serveur d'API, votre VPC doit avoir `enableDnsHostnames` et `enableDnsSupport` définis sur `true`, et les options DHCP définies pour votre VPC doivent inclure `AmazonProvidedDNS` dans leur liste de serveurs de nom de domaine. Pour plus d'informations, consultez Mise à jour du support DNS pour votre VPC dans le Guide de l'utilisateur d'Amazon VPC.

Vous pouvez définir les exigences d'accès au point de terminaison de votre serveur d'API lorsque vous créez un nouveau cluster, et vous pouvez mettre à jour l'accès au point de terminaison du serveur d'API pour un cluster à tout moment.

Modification de l'accès au point de terminaison de cluster

Utilisez les procédures de cette section afin de modifier l'accès au point de terminaison pour un cluster existant. Le tableau suivant présente les combinaisons d'accès au point de terminaison de serveur d'API prises en charge et leur comportement.

Accès public au point de terminaison	Accès privé au point de terminaison	Attitude
Activé	Désactivées	• Il s'agit du comportement par défaut pour les nouveaux clusters Amazon EKS. • Les demandes d'API Kubernetes qui proviennent du VPC de votre cluster (telles que la communication entre le nœud et le plan de contrôle) quittent le VPC mais pas le réseau d'Amazon. • Le serveur d'API de votre cluster est accessible depuis internet. Si nécessaire, vous pouvez limiter les blocs CIDR qui peuvent accéder au point de terminais on public. Si vous limitez l'accès à des blocs CIDR spécifiques, il est recommandé d'activer également le point de terminaison privé ou de vous assurer que les blocs CIDR que vous spécifiez incluent les adresses à partir desquelles les nœuds et les pods Fargate (si vous les utilisez) accèdent au point de terminaison public.
Activées	Activées	• Les demandes d'API Kubernetes au sein du VPC de votre cluster (telles que la communication entre le nœud et le plan de contrôle) utilisent le point de terminaison VPC privé. • Le serveur d'API de votre cluster est accessible depuis internet. Si nécessaire, vous pouvez limiter les blocs CIDR qui peuvent accéder au point de terminaison public.

Accès public au point de terminaison	Accès privé au point de terminaison	Attitude
		• Si vous utilisez des nœuds hybrides avec votre cluster Amazon EKS, il n'est pas recommandé d'activer à la fois l'accès aux points de terminaison du cluster public et privé. Comme vos nœuds hybrides s'exécute nt en dehors de votre VPC, ils relieront le point de terminaison du cluster aux adresses IP publiques. Il est recommandé d'utiliser un accès aux points de terminais on de cluster public ou privé pour les clusters dotés de nœuds hybrides.

Désactivées	Activées	• Tout le trafic vers le serveur API de votre cluster doit provenir du VPC de votre cluster ou d'un réseau connecté.
		• Il n'y a pas d'accès public vers votre serveur d'API depuis Internet. Toutes les commandes `kubectl` doivent provenir du VPC ou d'un réseau connecté. Pour les options de connectivité, consultez the section called "Accès à un serveur d'API privé uniquement".
		• Le point de terminaison du serveur API du cluster est résolu par les serveurs DNS publics en une adresse IP privée provenant du VPC. Dans le passé, le point de terminaison ne pouvait être résolu qu'à partir du VPC.
		
		Si votre point de terminaison ne se résout pas à une adresse IP privée dans le VPC pour un cluster existant, vous pouvez :
		  • Activer l'accès public, puis le désactiver à nouveau. Vous n'avez besoin de le faire qu'une seule fois pour un cluster et le point de terminaison se résoudra en une adresse IP privée à partir de ce point.
		  • Mettez à jour votre cluster.

Blocs CIDR dans le point de terminaison public (**`IPv6`**cluster)

Vous pouvez ajouter `IPv6` des blocs `IPv4` CIDR au point de terminaison public d'un `IPv6` cluster, car le point de terminaison public est à double pile. Cela ne s'applique `IPv6` qu'aux nouveaux clusters `ipFamily` définis en octobre 2024 ou ultérieurement. Vous pouvez identifier ces clusters par le nouveau nom de domaine du point de terminaison`api.aws`.

Blocs CIDR dans le point de terminaison public (**`IPv4`**cluster)

Vous pouvez ajouter des blocs `IPv4` CIDR au point de terminaison public d'un `IPv4` cluster. Vous ne pouvez pas ajouter de blocs `IPv6` CIDR au point de terminaison public d'un `IPv4` cluster. Si vous essayez, EKS renvoie le message d'erreur suivant : `The following CIDRs are invalid in publicAccessCidrs`

Blocs CIDR dans le point de terminaison public (**`IPv6`**cluster créé avant octobre 2024)

Vous pouvez ajouter des blocs `IPv4` CIDR au point de terminaison public des anciens `IPv6` clusters que vous avez créés avant octobre 2024. Vous pouvez identifier ces clusters par le `eks.amazonaws.com` point de terminaison. Vous ne pouvez pas ajouter de blocs `IPv6` CIDR au point de terminaison public de ces anciens `IPv6` clusters que vous avez créés avant octobre 2024. Si vous essayez, EKS renvoie le message d'erreur suivant : `The following CIDRs are invalid in publicAccessCidrs`

Accès à un serveur d'API privé uniquement

Si vous avez désactivé l'accès public pour le point de terminaison du serveur d'API Kubernetes de votre cluster, vous ne pouvez accéder au serveur d'API que depuis votre VPC ou un réseau connecté. Voici quelques méthodes d'accès possibles au point de terminaison du serveur d'API Kubernetes :

Réseau connecté

Connectez votre réseau au VPC à l'aide d'une passerelle de transit AWS ou d'une autre option de connectivité, puis utilisez un ordinateur dans le réseau connecté. Vous devez vous assurer que le groupe de sécurité de votre plan de contrôle Amazon EKS contient des règles pour autoriser le trafic entrant sur le port 443 depuis votre réseau connecté.

Hôte Amazon EC2 Bastion

Vous pouvez lancer une EC2 instance Amazon dans un sous-réseau public du VPC de votre cluster, puis vous connecter via SSH à cette instance pour exécuter des commandes. `kubectl`

Pour plus d'informations, consultez la section Hôtes bastions Linux sur AWS. Vous devez vous assurer que le groupe de sécurité de votre plan de contrôle Amazon EKS contient des règles pour autoriser le trafic entrant sur le port 443 depuis l'hôte bastion. Pour de plus amples informations, veuillez consulter the section called "Exigences de groupe de sécurité".

Lorsque vous configurez `kubectl` votre hôte bastion, veillez à utiliser des AWS informations d'identification déjà mappées à la configuration RBAC de votre cluster, ou ajoutez le principal IAM que votre bastion utilisera à la configuration RBAC avant de supprimer l'accès public du point de terminaison. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Accès non autorisé ou refusé (`kubectl`)".

AWS Environnement de développement intégré Cloud9

AWS Cloud9 est un environnement de développement intégré (IDE) basé sur le cloud qui vous permet d'écrire, d'exécuter et de déboguer votre code avec un simple navigateur. Vous pouvez créer un IDE AWS Cloud9 dans le VPC de votre cluster et utiliser l'IDE pour communiquer avec votre cluster. Pour plus d'informations, consultez Création d'un environnement dans AWS Cloud9. Vous devez vous assurer que votre groupe de sécurité de plan de contrôle Amazon EKS contient des règles permettant d'autoriser le trafic entrant sur le port 443 à partir de votre groupe de sécurité IDE. Pour de plus amples informations, veuillez consulter the section called "Exigences de groupe de sécurité".

Lorsque vous configurez `kubectl` votre IDE AWS Cloud9, veillez à utiliser des AWS informations d'identification déjà mappées à la configuration RBAC de votre cluster, ou ajoutez le principal IAM que votre IDE utilisera à la configuration RBAC avant de supprimer l'accès public des terminaux. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Accès non autorisé ou refusé (`kubectl`)".

Modifiez cette page sur GitHub

Configuration de l'accès réseau au point de terminaison du serveur API du cluster

Vous pouvez modifier l'accès aux points de terminaison de votre serveur API de cluster à l'aide de la AWS CLI AWS Management Console ou dans les sections suivantes.

Configuration de l'accès aux terminaux - AWS console

1. Ouvrez la console Amazon EKS.

2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Choisissez l'onglet Mise en réseau, puis sélectionnez Gérer l'accès aux terminaux.
4. Pour l'accès privé, choisissez d'activer ou de désactiver l'accès privé pour le point de terminaison du serveur d'KubernetesAPI de votre cluster. Si vous activez l'accès privé, les demandes d'KubernetesAPI provenant du VPC de votre cluster utilisent le point de terminaison VPC privé. Vous devez activer l'accès privé pour désactiver l'accès public.
5. Pour l'accès public, choisissez d'activer ou de désactiver l'accès public pour le point de terminaison du serveur d'KubernetesAPI de votre cluster. Si vous désactivez l'accès public, le serveur Kubernetes API de votre cluster ne peut recevoir des demandes que depuis le VPC du cluster.
6. (Facultatif) Si vous avez activé l'accès public, vous pouvez spécifier les adresses Internet qui peuvent communiquer avec le point de terminaison public. Sélectionnez Advanced Settings (Paramètres avancés). Entrez un bloc CIDR, tel que *203.0.113.5/32*. Le bloc ne peut pas inclure d'adresses réservées. Vous pouvez entrer des blocs supplémentaires en sélectionnant Add Source (Ajouter une source). Vous pouvez spécifier un nombre maximal de blocs CIDR. Pour de plus amples informations, veuillez consulter the section called "Quotas de service". Si vous ne spécifiez aucun bloc, le point de terminaison du serveur d'API public reçoit des demandes de toutes les adresses IP pour `IPv4` (`0.0.0.0/0`) et en plus `IPv6` (`::/0`) pour le `IPv6` cluster à double pile. Si vous limitez l'accès à votre point de terminaison public à l'aide de blocs CIDR, nous vous recommandons d'activer également l'accès au point de terminaison privé afin que les nœuds et Pods Fargate (si vous les utilisez) puissent communiquer avec le cluster. Sans le point de terminaison privé activé, vos sources CIDR de point d'accès public doivent inclure les sources de sortie de votre VPC. Par exemple, si vous avez un nœud dans un sous-réseau privé qui communique à Internet via une passerelle NAT, vous devez ajouter l'adresse IP sortante de la passerelle NAT dans le cadre d'un bloc CIDR autorisé sur votre point de terminaison public.
7. Choisissez Update (Mettre à jour) pour terminer.

Configurer l'accès aux points de terminaison - AWS CLI

Effectuez les étapes suivantes à l'aide de la version AWS CLI `1.27.160` ou ultérieure. Vous pouvez vérifier votre version actuelle avec `aws --version`. Pour installer ou mettre à niveau la AWS CLI, reportez-vous à la section Installation de la AWS CLI.

1. Mettez à jour l'accès aux points de terminaison de votre serveur d'API de cluster à l'aide de la commande AWS CLI suivante. Remplacez les valeurs par le nom de votre cluster et les valeurs d'accès au point de terminaison souhaitées. Si vous définissez `endpointPublicAccess=true`,

vous pouvez (éventuellement) entrer un seul bloc CIDR ou une liste de blocs CIDR séparés par des virgules pour `publicAccessCidrs`. Les blocs ne peuvent pas inclure d'adresses réservées. Si vous spécifiez des blocs CIDR, le point de terminaison du serveur d'API public ne recevra que les demandes des blocs répertoriés. Vous pouvez spécifier un nombre maximal de blocs CIDR. Pour de plus amples informations, veuillez consulter the section called "Quotas de service". Si vous limitez l'accès à votre point de terminaison public à l'aide de blocs CIDR, il est recommandé d'activer également l'accès au point de terminaison privé afin que les nœuds et les Pods Fargate (si vous les utilisez) puissent communiquer avec le cluster. Sans le point de terminaison privé activé, vos sources CIDR de point d'accès public doivent inclure les sources de sortie de votre VPC. Par exemple, si vous avez un nœud dans un sous-réseau privé qui communique à Internet via une passerelle NAT, vous devez ajouter l'adresse IP sortante de la passerelle NAT dans le cadre d'un bloc CIDR autorisé sur votre point de terminaison public. Si vous ne spécifiez aucun bloc CIDR, le point de terminaison du serveur d'API public reçoit les demandes de toutes les adresses IP (0.0.0.0/0) et en plus `IPv6` (`::/0`) pour le cluster à double pile. `IPv6`

> **Note**
>
> La commande suivante active l'accès privé et l'accès public à partir d'une adresse IP unique pour le point de terminaison du serveur API. Remplacez *`203.0.113.5/32`* par un seul bloc CIDR ou une liste de blocs CIDR séparés par des virgules auxquels vous souhaitez restreindre l'accès réseau.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
```

```
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. Surveillez le statut de la mise à jour de l'accès à votre point de terminaison avec la commande suivante, en indiquant le nom de votre cluster et l'ID de mise à jour qui a été renvoyé par la commande précédente. Votre mise à jour est terminée lorsqu'elle affiche l'état `Successful`.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
```

```
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

Modifiez cette page sur GitHub

Déployer des nœuds Windows sur des clusters EKS

Découvrez comment activer et gérer le support Windows pour votre cluster Amazon EKS afin d'exécuter des conteneurs Windows parallèlement à des conteneurs Linux.

Considérations

Avant de déployer des nœuds Windows, veuillez tenir compte des considérations suivantes.

- Le mode automatique EKS ne prend pas en charge les nœuds Windows
- Vous pouvez utiliser le réseau hôte sur les nœuds Windows à l'aide des pods `HostProcess`. Pour plus d'informations, consultez la section Créer une fenêtre HostProcessPod dans la documentation de Kubernetes.
- Les clusters Amazon EKS doivent contenir un ou plusieurs nœuds Linux ou Fargate pour exécuter les pods du système principal qui s'exécutent uniquement sous Linux, tels que CoreDNS.
- Les journaux d'`kube-proxy`événements `kubelet` et les journaux d'`EKS Windows`événements sont redirigés vers le journal des événements et leur limite est fixée à 200 Mo.
- Vous ne pouvez pas utiliser le même rôle IAM pour les nœuds Linux et Windows.
- Vous ne pouvez pas utiliser l'option Attribuer des groupes de sécurité à des pods individuels lorsque des pods s'exécutent sur des nœuds Windows.
- Vous ne pouvez pas utiliser le réseau personnalisé avec des nœuds Windows.
- Vous ne pouvez pas l'utiliser `IPv6` avec des nœuds Windows.
- Les nœuds Windows prennent en charge une interface réseau Elastic par nœud. Par défaut, le nombre de pods que vous pouvez exécuter par nœud Windows est égal au nombre d'adresses IP

disponibles par elastic network interface pour le type d'instance du nœud, moins une. Pour plus d'informations, consultez la section Adresses IP par interface réseau et par type d'instance dans le guide de EC2 l'utilisateur Amazon.

- Dans un cluster Amazon EKS, un seul service doté d'un équilibreur de charge peut prendre en charge jusqu'à 1 024 pods principaux. Chaque pod a sa propre adresse IP. La limite précédente de 64 pods n'est plus le cas, après une mise à jour de Windows Server à partir de la version 17763.2746 du système d'exploitation.
- Les conteneurs Windows ne sont pas pris en charge pour les pods Amazon EKS sur Fargate.
- Vous ne pouvez pas utiliser Amazon EKS Hybrid Nodes avec Windows comme système d'exploitation pour l'hôte.
- Vous ne pouvez pas récupérer les journaux depuis le `vpc-resource-controller` Pod. Vous le pouviez auparavant lorsque vous déployiez le contrôleur sur le plan de données.
- Il y a un temps de stabilisation avant qu'une adresse `IPv4` ne soit affectée à un nouveau pod. Cela empêche le trafic de s'écouler vers un pod plus ancien avec la même adresse `IPv4` en raison de règles `kube-proxy` périmées.
- La source du contrôleur est gérée sur GitHub. Pour contribuer ou signaler des problèmes contre le contrôleur, consultez le projet sur GitHub.
- Lorsque vous spécifiez un ID AMI personnalisé pour les groupes de nœuds gérés par Windows, ajoutez-le `eks:kube-proxy-windows` à votre carte de configuration AWS IAM Authenticator. Pour de plus amples informations, veuillez consulter the section called "Limites et conditions lors de la spécification d'un ID d'AMI".
- S'il est essentiel de préserver IPv4 les adresses disponibles pour votre sous-réseau, reportez-vous au Guide des meilleures pratiques d'EKS - Gestion des adresses IP réseau Windows pour obtenir des conseils.
- Considérations relatives aux entrées EKS Access
  - Les entrées d'accès destinées à être utilisées avec les nœuds Windows doivent être du type de`EC2_WINDOWS`. Pour de plus amples informations, veuillez consulter the section called "Création d'entrées d'accès".

    Pour créer une entrée d'accès pour un nœud Windows :

    ```
    aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
     iam::111122223333:role/<role-name> --type EC2_Windows
    ```

Prérequis

- Un cluster existant.
- Votre cluster doit avoir au moins un (nous recommandons au moins deux) nœud Linux ou Fargate Pod pour exécuter CoreDNS. Si vous activez le support Windows d'ancienne génération, vous devez utiliser un nœud Linux (vous ne pouvez pas utiliser un Fargate Pod) pour exécuter CoreDNS.
- Rôle IAM existant dans le cluster Amazon EKS.

Activer le support Windows

1. Si votre cluster ne contient pas de nœuds Amazon Linux et que vous utilisez des groupes de sécurité pour les pods, passez à l'étape suivante. Sinon, confirmez que la politique gérée `AmazonEKSVPCResourceController` est attachée à votre rôle de cluster. Remplacez *`eksClusterRole`* par le nom de rôle de votre cluster.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   L'exemple qui suit illustre un résultat.

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSClusterPolicy"
           },
           {
               "PolicyName": "AmazonEKSVPCResourceController",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSVPCResourceController"
           }
       ]
   }
   ```

   Si la politique est attachée, comme c'est le cas dans la sortie précédente, passez à l'étape suivante.

2. Associez la politique gérée par Amazon EKSVPCResource Controller au rôle IAM de votre cluster Amazon EKS. Remplacez *`eksClusterRole`* par le nom de rôle de votre cluster.

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
```

3. Mettez à jour le VPC CNI ConfigMap pour activer Windows IPAM. Notez que si le VPC CNI est installé sur votre cluster à l'aide d'un graphique Helm ou en tant que module complémentaire Amazon EKS, vous ne pourrez peut-être pas modifier directement le. ConfigMap Pour plus d'informations sur la configuration des modules complémentaires Amazon EKS, consultezthe section called "Champs que vous pouvez personnaliser".

   a. Créez un fichier nommé *vpc-resource-controller-configmap.yaml* avec les contenus suivants.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: amazon-vpc-cni
     namespace: kube-system
   data:
     enable-windows-ipam: "true"
   ```

   b. Appliquer le `ConfigMap` à votre cluster.

   ```
   kubectl apply -f vpc-resource-controller-configmap.yaml
   ```

4. Si le mode d'authentification de votre cluster est configuré pour activer le `aws-auth` configmap :

   - Vérifiez que vous disposez `aws-auth` `ConfigMap` d'un mappage pour le rôle d'instance du nœud Windows afin d'inclure le groupe d'autorisations `eks:kube-proxy-windows` RBAC. Vous pouvez vérifier en exécutant la commande suivante.

     ```
     kubectl get configmap aws-auth -n kube-system -o yaml
     ```

     L'exemple qui suit illustre un résultat.

     ```
     apiVersion: v1
     kind: ConfigMap
     metadata:
       name: aws-auth
       namespace: kube-system
     data:
     ```

```
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws: iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

Vous devriez voir `eks:kube-proxy-windows` répertorié sous les groupes. Si le groupe n'est pas spécifié, vous devez le mettre à jour `ConfigMap` ou le créer pour inclure le groupe requis. Pour en savoir plus sur `ConfigMap aws-auth`, consultez the section called "Appliquer la `ConfigMap aws-auth` à votre cluster".

5. Si le mode d'authentification de votre cluster est configuré pour désactiver le `aws-auth` configmap, vous pouvez utiliser les entrées d'accès EKS. Créez un nouveau rôle de nœud à utiliser avec les instances Windows, et EKS créera automatiquement une entrée d'accès de ce type`EC2_WINDOWS`.

Déployer des modules Windows

Lorsque vous déployez des pods sur votre cluster, vous devez spécifier le système d'exploitation qu'ils utilisent si vous exécutez plusieurs types de nœuds.

Pour les modules Linux, utilisez le texte de sélection de nœuds suivant dans vos manifestes.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Pour les modules Windows, utilisez le texte de sélection de nœuds suivant dans vos manifestes.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

Vous pouvez déployer un exemple d'application pour voir les sélecteurs de nœuds utilisés.

Support d'une densité de pods plus élevée sur les nœuds Windows

Dans Amazon EKS, une `IPv4` adresse est attribuée à chaque pod par votre VPC. De ce fait, le nombre de pods que vous pouvez déployer sur un nœud est limité par les adresses IP disponibles, même si les ressources sont suffisantes pour exécuter davantage de pods sur le nœud. Étant donné qu'une seule interface réseau élastique est prise en charge par un nœud Windows, par défaut, le nombre maximal d'adresses IP disponibles sur un nœud Windows est égal à :

```
Number of private IPv4 addresses for each interface on the node - 1
```

Une adresse IP est utilisée comme adresse IP principale de l'interface réseau, elle ne peut donc pas être allouée aux Pods.

Vous pouvez augmenter la densité des pods sur les nœuds Windows en activant la délégation de préfixes IP. Cette fonctionnalité vous permet d'attribuer un préfixe `/28` `IPv4` à l'interface réseau principale, au lieu d'attribuer des adresses secondaires `IPv4`. L'attribution d'un préfixe IP augmente le nombre maximum d'adresses `IPv4` disponibles sur le nœud à :

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

Avec ce nombre nettement plus élevé d'adresses IP disponibles, les adresses IP disponibles ne devraient pas limiter votre capacité à augmenter le nombre de pods sur vos nœuds. Pour de plus amples informations, veuillez consulter the section called “Augmenter le nombre d'adresses IP”.

Désactiver le support Windows

1. Si votre cluster contient des nœuds Amazon Linux et que vous utilisez des groupes de sécurité pour les pods, ignorez cette étape.

   Supprimez la politique IAM gérée `AmazonVPCResourceController` de votre rôle de cluster. *`eksClusterRole`*Remplacez-le par le nom de votre rôle de cluster.

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. Désactivez Windows IPAM dans le. `amazon-vpc-cni` ConfigMap

```
kubectl patch configmap/amazon-vpc-cni \
                    -n kube-system \
                    --type merge \
                    -p '{"data":{"enable-windows-ipam":"false"}}'
```

Déployez des clusters privés avec un accès Internet limité

Cette rubrique décrit comment déployer un cluster Amazon EKS déployé sur le AWS cloud, mais ne disposant pas d'un accès Internet sortant. Si vous avez un cluster local sur AWS Outposts, consultezthe section called "Nœuds", au lieu de cette rubrique.

Si vous n'êtes pas familier avec le réseau Amazon EKS, consultez Démystifier le réseau de clusters pour les nœuds de travail Amazon EKS. Si votre cluster ne dispose pas d'un accès Internet sortant, il doit répondre aux exigences suivantes :

- Votre cluster doit extraire des images d'un registre de conteneurs qui se trouve dans votre VPC. Vous pouvez créer un registre de conteneurs Amazon Elastic Container Registry dans votre VPC et y copier des images de conteneurs pour que vos nœuds puissent les extraire. Pour de plus amples informations, veuillez consulter the section called "Copier une image dans un référentiel".
- Votre cluster doit avoir un accès privé au point de terminaison activé. Ceci est nécessaire pour que les nœuds s'enregistrent auprès du point de terminaison du cluster. L'accès public au point de terminaison est facultatif. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".
- Les nœuds Linux et Windows autogérés doivent inclure les arguments bootstrap suivants avant d'être lancés. Ces arguments contournent l'introspection d'Amazon EKS et ne nécessitent pas d'accès à l'API Amazon EKS depuis le VPC.

  a. Déterminez la valeur du point de terminaison de votre cluster à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
  ```

  L'exemple qui suit illustre un résultat.

  ```
  https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
  ```

b. Déterminez la valeur de l'autorité de certification de votre cluster à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --
output text
```

La sortie renvoyée est une longue chaîne.

c. Remplacez *cluster-endpoint* et *certificate-authority* dans les commandes suivantes par les valeurs renvoyées dans la sortie des commandes précédentes. Pour plus d'informations sur la spécification des arguments d'amorçage lors du lancement de nœuds autogérés, consultez the section called "Amazon Linux" et the section called "Windows".

- Pour les nœuds Linux :

```
--apiserver-endpoint cluster-endpoint --b64-cluster-ca certificate-authority
```

Pour des arguments supplémentaires, consultez le script bootstrap sur GitHub.

- Pour les nœuds Windows :

> **Note**
>
> Si vous utilisez un service CIDR personnalisé, vous devez le spécifier à l'aide du `-ServiceCIDR` paramètre. Dans le cas contraire, la résolution DNS des pods du cluster échouera.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Pour des arguments supplémentaires, consultez the section called "Paramètres de configuration du script d'amorçage".

- Votre cluster `aws-auth` `ConfigMap` doit être créé depuis votre VPC. Pour plus d'informations sur la création et l'ajout d'entrées dans la `ConfigMap` `aws-auth`, entrez `eksctl create iamidentitymapping --help` dans votre terminal. S'il `ConfigMap` n'existe pas sur votre serveur, il le `eksctl` créera lorsque vous utiliserez la commande pour ajouter un mappage d'identité.

- Les pods configurés avec des rôles IAM pour les comptes de service obtiennent des informations d'identification à partir d'un appel d'API du AWS Security Token Service (AWS STS). S'il n'y a pas d'accès Internet sortant, vous devez créer et utiliser un point de terminaison VPC AWS STS dans votre VPC. La plupart AWS `v1` SDKs utilisent le point de terminaison AWS STS global par défaut (`sts.amazonaws.com`), qui n'utilise pas le point de terminaison VPC AWS STS. Pour utiliser le point de terminaison VPC AWS STS, vous devrez peut-être configurer votre SDK pour utiliser le point de terminaison AWS STS régional (). `sts.`*`region-code`*`.amazonaws.com` Pour de plus amples informations, veuillez consulter the section called "Points de terminaison STS".
- Les sous-réseaux VPC de votre cluster doivent disposer d'un point de terminaison d'interface VPC pour tous les AWS services auxquels vos pods doivent accéder. Pour plus d'informations, consultez Accéder à un AWS service à l'aide d'un point de terminaison VPC d'interface. Certains services et points de terminaison couramment utilisés sont répertoriés dans le tableau suivant. Pour une liste complète des points de terminaison, consultez Services AWS qui s'intègrent à AWS PrivateLink dans le Guide AWS PrivateLink .

  Nous vous recommandons d'activer les noms DNS privés pour vos points de terminaison VPC, afin que les charges de travail puissent continuer à utiliser les points de terminaison de AWS service public sans problème.

Service	Point de terminaison
Amazon EC2	com.amazonaws. *region-code* .ec2
Amazon Elastic Container Registry (pour extraire des images de conteneurs)	com.amazonaws. *region-code* .ecr.api, com.amazonaws. *region-code* .ecr.dkr et com.amazonaws. *region-code* .s3
Équilibreurs de charge Application Load Balancer et Network Load Balancer	com.amazonaws. *region-code* . équilibrage de charge élastique
AWS X-Ray	com.amazonaws. *region-code* .xray
Amazon CloudWatch Logs	com.amazonaws. *region-code* .journaux
AWS Service de jeton de sécurité (obligatoire lors de l'utilisation de rôles IAM pour les comptes de service)	com.amazonaws. *region-code* .sts

Service	Point de terminaison
Amazon EKS Auth (obligatoire lors de l'utilisation des associations Pod Identity)	com.amazonaws. *region-code* .eks-auth
Amazon EKS	com.amazonaws. *region-code* .eks

- Tous les nœuds autogérés doivent être déployés sur des sous-réseaux qui disposent des points de terminaison de l'interface VPC dont vous avez besoin. Si vous créez un groupe de nœuds gérés, le groupe de sécurité des points de terminaison de l'interface VPC doit autoriser le CIDR pour les sous-réseaux, ou vous devez ajouter le groupe de sécurité des nœuds créé au groupe de sécurité des points de terminaison de l'interface VPC.
- Si vos pods utilisent des volumes Amazon EFS, avant de déployer le système de fichiers Store an Elastic avec Amazon EFS, le fichier kustomization.yaml du pilote doit être modifié pour que les images du conteneur utilisent la AWS même région que le cluster Amazon EKS.
- Route53 ne prend pas en charge. AWS PrivateLink Vous ne pouvez pas gérer les enregistrements DNS Route53 à partir d'un cluster Amazon EKS privé. Cela a un impact sur le DNS externe de Kubernetes.
- Si vous utilisez l'AMI optimisée EKS, vous devez activer le `ec2` point de terminaison dans le tableau ci-dessus. Vous pouvez également définir manuellement le nom DNS du nœud. L'AMI optimisée permet EC2 APIs de définir automatiquement le nom DNS du nœud.
- Vous pouvez utiliser le AWS Load Balancer Controller pour déployer des équilibreurs de charge AWS d'application (ALB) et des équilibreurs de charge réseau sur votre cluster privé. Lors de son déploiement, vous devez utiliser les indicateurs de ligne de commande pour définir `enable-shield`, `enable-waf` et `enable-wafv2` sur false. La découverte de certificats à l'aide de noms d'hôtes provenant d'objets Ingress n'est pas prise en charge. Cela est dû au fait que le contrôleur doit atteindre AWS Certificate Manager, qui ne possède pas de point de terminaison d'interface VPC.

  Le contrôleur prend en charge les Network Load Balancers avec des cibles IP, qui sont nécessaires pour une utilisation avec Fargate. Pour plus d'informations, consultez the section called "Répartition de la charge des applications" et the section called "Créer un équilibreur de charge de réseau".
- Cluster Autoscaler est pris en charge. Lorsque vous déployez des pods Cluster Autoscaler, assurez-vous que la ligne de commande inclut. `--aws-use-static-instance-list=true` Pour plus d'informations, consultez la section Utiliser une liste d'instances statiques sur GitHub.

Le VPC du nœud de travail doit également inclure le point de terminaison VPC STS et le point de AWS terminaison VPC à mise à l'échelle automatique.

- Certains logiciels de conteneur utilisent des appels d'API qui accèdent au AWS Marketplace Metering Service pour surveiller l'utilisation. Les clusters privés n'autorisent pas ces appels. Vous ne pouvez donc pas utiliser ces types de conteneurs dans des clusters privés.

Faites évoluer le calcul en cluster avec Karpenter et Cluster Autoscaler

La scalabilité automatique est une fonction qui augmente ou diminue automatiquement vos ressources en fonction de l'évolution de la demande. Il s'agit d'une fonction majeure de Kubernetes qui, autrement, nécessiterait des ressources humaines importantes pour être exécutée manuellement.

Mode automatique EKS

Le mode automatique Amazon EKS adapte automatiquement les ressources de calcul du cluster. Si un module ne peut pas être installé sur des nœuds existants, le mode automatique EKS en crée un nouveau. Le mode automatique EKS consolide également les charges de travail et supprime les nœuds. Le mode automatique d'EKS s'appuie sur Karpenter.

Pour plus d'informations, consultez :

- Mode automatique EKS
- the section called "Créer un pool de nœuds"
- the section called "Déployez une charge de travail excessive"

Solutions supplémentaires

Amazon EKS prend en charge deux produits de mise à l'échelle automatique supplémentaires :

Charpentier

Karpenter est un autoscaler de cluster Kubernetes flexible et performant qui permet d'améliorer la disponibilité des applications et l'efficacité du cluster. Karpenter lance des ressources de calcul adaptées (par exemple, des EC2 instances Amazon) en réponse à l'évolution de la charge de

l'application en moins d'une minute. En intégrant Kubernetes à Kubernetes AWS, Karpenter peut fournir des ressources de just-in-time calcul qui répondent précisément aux exigences de votre charge de travail. Karpenter alloue automatiquement de nouvelles ressources de calcul en fonction des exigences spécifiques des charges de travail du cluster. Celles-ci comprennent les exigences en matière de calcul, de stockage, d'accélération et de planification. Amazon EKS prend en charge les clusters utilisant Karpenter, bien que Karpenter fonctionne avec tout cluster Kubernetes conforme. Pour en savoir plus, veuillez consulter la documentation Karpenter.

> **Important**
>
> Karpenter est un logiciel open source que les AWS clients sont chargés d'installer, de configurer et de gérer dans leurs clusters Kubernetes. AWS fournit un support technique lorsque Karpenter est exécuté sans modification à l'aide d'une version compatible dans les clusters Amazon EKS. Il est essentiel que les clients maintiennent la disponibilité et la sécurité du contrôleur Karpenter ainsi que les procédures de test appropriées lors de sa mise à niveau ou du cluster Kubernetes dans lequel il s'exécute, comme tout autre logiciel géré par le client. Il n'existe aucun accord de niveau de AWS service (SLA) pour Karpenter et les clients sont tenus de s'assurer que les EC2 instances lancées par Karpenter répondent à leurs besoins commerciaux.

Autoscaleur de clusters

Le Kubernetes Cluster Autoscaler ajuste automatiquement le nombre de nœuds dans votre cluster lorsque les pods échouent ou sont reprogrammés sur d'autres nœuds. Le Cluster Autoscaler utilise des groupes Auto Scaling. Pour plus d'informations, consultez Cluster Autoscaler activé. AWS

En savoir plus sur le changement de zone d'Amazon Application Recovery Controller (ARC) dans Amazon EKS

Kubernetes possède des fonctionnalités natives qui vous permettent de rendre vos applications plus résilientes face à des événements tels que la détérioration de l'état de santé ou la détérioration d'une zone de disponibilité (AZ). Lorsque vous exécutez vos charges de travail dans un cluster Amazon EKS, vous pouvez améliorer encore la tolérance aux pannes de votre environnement applicatif et la restauration des applications grâce au changement de zone ou à l'autoshift de zone d'Amazon Application Recovery Controller (ARC). Le décalage de zone ARC est conçu pour être une mesure

temporaire qui vous permet de déplacer le trafic vers une ressource hors d'une zone AZ altérée jusqu'à ce que le décalage de zone expire ou que vous l'annuliez. Vous pouvez étendre le décalage de zone si nécessaire.

Vous pouvez démarrer un changement de zone pour un cluster EKS, ou vous pouvez autoriser le changement de zone AWS à votre place en activant le changement automatique de zone. Ce changement met à jour le flux de trafic east-to-west réseau dans votre cluster afin de ne considérer que les points de terminaison réseau des pods exécutés sur des nœuds de travail comme sains AZs. En outre, tout ALB ou NLB gérant le trafic entrant pour les applications de votre cluster EKS acheminera automatiquement le trafic vers les cibles saines. AZs Pour les clients qui recherchent les objectifs de disponibilité les plus élevés, en cas de défaillance d'une zone de disponibilité, il peut être important de pouvoir orienter tout le trafic vers cette zone jusqu'à ce qu'elle se rétablisse. Pour cela, vous pouvez également activer un ALB ou un NLB avec décalage de zone ARC.

Comprendre le flux de trafic réseau est-ouest entre les modules

Le schéma suivant illustre deux exemples de charges de travail : les commandes et les produits. Le but de cet exemple est de montrer comment les charges de travail et les pods AZs communiquent entre eux dans différents environnements.

AZ 1
Kubernetes Node
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
CoreDNS
Pod
Orders-abc
192.168.151.208
1
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
iptables
2
Update IP address
Packet
Src: 192.168.151.208
Dest: 192.168.170.82
ENI

1. Pour que les commandes puissent communiquer avec les produits, elles doivent d'abord résoudre le nom DNS du service de destination. Les commandes communiqueront avec CoreDNS pour récupérer l'adresse IP virtuelle (IP du cluster) pour ce service. Une fois que Orders a résolu le nom du service Products, il envoie le trafic vers cette adresse IP cible.
2. Le kube-proxy s'exécute sur tous les nœuds du cluster et surveille en permanence EndpointSlicesles services. Lorsqu'un service est créé, un EndpointSlice est créé et géré en arrière-plan par le EndpointSlice contrôleur. Chacun EndpointSlice possède une liste ou un tableau de points de terminaison contenant un sous-ensemble d'adresses Pod ainsi que les nœuds sur lesquels ils s'exécutent. Le kube-proxy définit des règles de routage pour chacun de ces points de terminaison Pod en utilisant `iptables` les nœuds. Le kube-proxy est également responsable d'une forme de base d'équilibrage de charge en redirigeant le trafic destiné à l'adresse IP du cluster d'un service pour qu'il soit envoyé directement à l'adresse IP d'un pod. Pour ce faire, le kube-proxy réécrit l'adresse IP de destination sur la connexion sortante.
3. Les paquets réseau sont ensuite envoyés au Products Pod dans AZ 2 via ENIs les nœuds respectifs (comme illustré dans le schéma ci-dessus).

Comprendre le changement de zone ARC dans Amazon EKS

En cas d'altération de l'AZ dans votre environnement, vous pouvez initier un changement de zone pour votre environnement de cluster EKS. Vous pouvez également autoriser la gestion AWS de cela pour vous grâce à l'autoshift zonal. Grâce à l'autoshift zonal, il AWS surveillera l'état général de l'AZ et répondra à une éventuelle détérioration de l'AZ en transférant automatiquement le trafic de l'AZ altéré dans votre environnement de cluster.

Une fois que le décalage de zone est activé avec ARC sur votre cluster Amazon EKS, vous pouvez déclencher un changement de zone ou activer le décalage automatique de zone à l'aide de la console ARC, de la AWS CLI ou du décalage zonal et du décalage automatique de zone. APIs Lors d'un changement de zone EKS, les opérations suivantes se produiront automatiquement :

- Tous les nœuds de l'AZ concernée seront bouclés. Cela empêchera le planificateur Kubernetes de planifier de nouveaux pods sur les nœuds de l'AZ malsaine.
- Si vous utilisez des groupes de nœuds gérés, le rééquilibrage des zones de disponibilité sera suspendu et votre Auto Scaling Group (ASG) sera mis à jour pour garantir que les nouveaux nœuds EKS Data Plane ne soient lancés que s'ils sont sains. AZs
- Les nœuds de l'AZ en mauvais état ne seront pas résiliés et les pods ne seront pas expulsés de ces nœuds. Cela permet de garantir qu'en cas d'expiration ou d'annulation d'un changement de zone, votre trafic puisse être renvoyé en toute sécurité vers l'AZ, qui est toujours à pleine capacité
- Le EndpointSlice contrôleur trouvera tous les points de terminaison du Pod dans la zone AZ altérée et les retirera de la zone correspondante EndpointSlices. Cela garantira que seuls les points de terminaison Pod sains AZs sont ciblés pour recevoir du trafic réseau. Lorsqu'un changement de zone est annulé ou expire, le EndpointSlice contrôleur le met à jour EndpointSlices pour inclure les points de terminaison dans l'AZ restaurée.

Les diagrammes ci-dessous illustrent de manière détaillée comment le changement de zone EKS garantit que seuls les points de terminaison Pod sains sont ciblés dans votre environnement de cluster.

AZ 1
Kubernetes Node
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
CoreDNS
Pod
Orders-abc
192.168.151.208
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
iptables
Update IP address
Packet
Src: 192.168.151.208
Dest: 192.168.170.81
ENI

Exigences d'EKS Zonal Shift

Pour que le changement de zone fonctionne correctement dans EKS, vous devez au préalable configurer votre environnement de cluster de manière à ce qu'il soit résilient à une altération de l'AZ. Vous trouverez ci-dessous une liste des étapes à suivre.

- Provisionnez les nœuds de travail de votre cluster sur plusieurs AZs
- Fournir une capacité de calcul suffisante pour résister à la suppression d'un seul AZ
- Prédimensionnez vos pods (y compris CoreDNS) dans chaque AZ
- Répartissez plusieurs répliques de Pod sur l'ensemble du AZs site pour vous assurer que le fait de vous éloigner d'un seul AZ vous permettra de disposer d'une capacité suffisante
- Colocalisez des pods interdépendants ou connexes dans la même zone
- Vérifiez que votre environnement de cluster fonctionnerait comme prévu avec un AZ de moins en démarrant manuellement un changement de zone. Vous pouvez également activer le changement automatique par zone et répondre aux essais d'entraînement du changement automatique. Cela

n'est pas nécessaire pour que le changement de zone fonctionne dans EKS, mais c'est fortement recommandé.

Provisionnez vos nœuds de travail EKS sur plusieurs AZs

AWS Les régions disposent de plusieurs sites distincts dotés de centres de données physiques appelés zones de disponibilité (AZs). AZs sont conçus pour être physiquement isolés les uns des autres afin d'éviter un impact simultané susceptible d'affecter une région entière. Lorsque vous provisionnez un cluster EKS, vous devez déployer vos nœuds de travail sur plusieurs nœuds au AZs sein d'une même région. Cela rendra votre environnement de cluster plus résilient face à la détérioration d'une seule AZ et vous permettra de maintenir la haute disponibilité (HA) de vos applications exécutées dans l'autre AZs. Lorsque vous commencez un changement de zone par rapport à la zone de zone affectée, le réseau intégré au cluster de votre environnement EKS se met automatiquement à jour pour n'utiliser que les zones saines AZs, tout en maintenant une position de haute disponibilité pour votre cluster.

Le fait de disposer d'une telle configuration multi-AZ pour votre environnement EKS améliorera la fiabilité globale de votre système. Cependant, les environnements multi-AZ peuvent jouer un rôle important dans la manière dont les données d'application sont transférées et traitées, ce qui aura un impact sur les frais de réseau de votre environnement. En particulier, le trafic sortant fréquent entre les zones (trafic réparti entre les zones AZs) peut avoir un impact majeur sur les coûts liés à votre réseau. Vous pouvez appliquer différentes stratégies pour contrôler le volume de trafic entre les zones entre les pods de votre cluster EKS et réduire les coûts associés. Reportez-vous à ce guide des meilleures pratiques pour plus de détails sur la manière d'optimiser les coûts du réseau lors de l'exécution d'environnements EKS à haute disponibilité.

Le schéma ci-dessous décrit un environnement EKS hautement disponible avec 3 environnements sains AZs.

Le schéma ci-dessous montre comment un environnement EKS avec 3 AZs est résilient à une déficience AZ et reste hautement disponible grâce aux 2 autres environnements sains AZs.

Fournir une capacité de calcul suffisante pour résister à la suppression d'un seul AZ

Pour optimiser l'utilisation des ressources et les coûts de votre infrastructure informatique dans le plan de données EKS, il est recommandé d'aligner la capacité de calcul sur les exigences de votre charge de travail. Toutefois, si tous vos nœuds de travail sont à pleine capacité, vous devez alors compter sur l'ajout de nouveaux nœuds de travail au plan de données EKS avant de pouvoir planifier de nouveaux pods. Lorsque vous exécutez des charges de travail critiques, il est généralement recommandé d'utiliser une capacité redondante en ligne pour faire face à des éventualités telles que des augmentations soudaines de la charge, des problèmes de santé des nœuds, etc. Si vous envisagez d'utiliser le décalage de zone, vous prévoyez de supprimer l'intégralité d'un AZ de capacité. Vous devez donc ajuster votre capacité de calcul redondante afin qu'elle soit suffisante pour supporter la charge même avec un AZ hors ligne.

Lors de la mise à l'échelle de votre capacité de calcul, le processus d'ajout de nouveaux nœuds au plan de données EKS prend un certain temps, ce qui peut avoir des répercussions sur les performances et la disponibilité en temps réel de vos applications, en particulier en cas de détérioration zonale. Votre environnement EKS doit être résilient pour absorber le fardeau lié à la perte d'un AZ afin d'éviter une dégradation de l'expérience pour vos utilisateurs finaux ou vos clients. Cela signifie minimiser ou éliminer tout décalage entre le moment où un nouveau Pod est nécessaire et le moment où il est réellement planifié sur un nœud de travail.

En outre, en cas d'altération de la zone, vous devez atténuer le risque d'une éventuelle contrainte de capacité de calcul qui empêcherait l'ajout de nouveaux nœuds à votre plan de données EKS en mode sain AZs.

Pour ce faire, vous devez surprovisionner la capacité de calcul de certains nœuds de travail de chacun des AZs afin que le Kubernetes Scheduler dispose d'une capacité préexistante pour les nouveaux emplacements de pods, en particulier lorsque vous avez un AZ de moins dans votre environnement.

Exécutez et répartissez plusieurs répliques de pods sur plusieurs AZs

Kubernetes vous permet de prédimensionner vos charges de travail en exécutant plusieurs instances (répliques Pod) d'une seule application. L'exécution de plusieurs répliques de Pod pour une application élimine un point de défaillance unique et augmente ses performances globales en réduisant la charge de ressources sur une seule réplique. Toutefois, pour garantir à la fois une haute disponibilité et une meilleure tolérance aux pannes pour vos applications, vous devez exécuter et répartir plusieurs répliques d'une application sur différents domaines de défaillance (également

appelés domaines de topologie) dans ce cas. AZs Grâce aux contraintes de dispersion topologique, vous pouvez configurer vos applications de manière à ce qu'elles présentent une stabilité statique préexistante afin que, en cas de défaillance de l'AZ, vous disposiez de suffisamment de répliques saines AZs pour gérer immédiatement tout pic ou augmentation de trafic supplémentaire qu'elles pourraient rencontrer.

Le schéma ci-dessous illustre un environnement EKS où le east-to-west trafic est fluide lorsque tout le monde est AZs en bonne santé.

Le schéma ci-dessous décrit un environnement EKS avec un flux de east-to-west trafic lorsqu'une seule AZ tombe en panne et que vous initiez un changement de zone.

L'extrait de code ci-dessous est un exemple de configuration de votre charge de travail avec cette fonctionnalité de Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders
spec:
  replicas: 9
  selector:
    matchLabels:
      app:orders
  template:
    metadata:
      labels:
        app: orders
        tier: backend
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
```

```
            app: orders
```

Plus important encore, vous devez exécuter plusieurs répliques de votre logiciel de serveur DNS (CoreDNS/Kube-DNS) et appliquer des contraintes de propagation topologique similaires si elles ne sont pas déjà configurées par défaut. Cela permettra de garantir que vous disposez d'un nombre suffisant de pods DNS sains pour continuer AZs à traiter les demandes de découverte de service pour les autres pods communicants du cluster en cas de défaillance d'AZ unique. Le module complémentaire CoreDNS EKS possède des paramètres par défaut pour que les pods CoreDNS soient répartis dans les zones de disponibilité de votre cluster si plusieurs nœuds sont disponibles. AZs Vous pouvez également remplacer ces paramètres par défaut par vos propres configurations personnalisées.

Lorsque vous installez CoreDNS avec Helm, vous pouvez mettre à jour `replicaCount` le fichier values.yaml pour vous assurer que vous disposez d'un nombre suffisant de répliques dans chaque AZ. En outre, pour vous assurer que ces répliques sont réparties entre les différents AZs environnements de votre cluster, vous devez mettre à jour la `topologySpreadConstraints` propriété dans le même fichier values.yaml. L'extrait de code ci-dessous montre comment configurer CoreDNS pour cela.

CoreDNS Helm values.yaml

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

En cas de défaillance de l'AZ, vous pouvez absorber la charge accrue sur les pods CoreDNS en utilisant un système de mise à l'échelle automatique pour CoreDNS. Le nombre d'instances DNS dont vous avez besoin dépend du nombre de charges de travail exécutées dans votre cluster. CoreDNS est lié au processeur, ce qui lui permet de s'adapter en fonction du processeur à l'aide de l'Horizontal Pod Autoscaler (HPA). Vous trouverez ci-dessous un exemple que vous pouvez modifier en fonction de vos besoins.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
```

```
  name: coredns
  namespace: default
spec:
  maxReplicas: 20
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  targetCPUUtilizationPercentage: 50
```

EKS peut également gérer le dimensionnement automatique du déploiement de CoreDNS dans la version complémentaire EKS de CoreDNS. Cet autoscaler CoreDNS surveille en permanence l'état du cluster, y compris le nombre de nœuds et de cœurs de processeur. Sur la base de ces informations, le contrôleur adaptera dynamiquement le nombre de répliques du déploiement CoreDNS dans un cluster EKS.

Pour activer la configuration de mise à l'échelle automatique dans le module complémentaire CoreDNS EKS, vous devez ajouter les paramètres de configuration facultatifs suivants :

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

Vous pouvez également utiliser le NodeLocal DNS ou l'autoscaler proportionnel au cluster pour dimensionner CoreDNS. Vous pouvez en savoir plus sur la mise à l'échelle horizontale de CoreDNS ici.

Colocalisez des pods interdépendants dans la même zone

Dans la plupart des cas, il se peut que vous exécutiez des charges de travail distinctes qui doivent communiquer entre elles pour mener à bien un end-to-end processus. Si les différentes applications sont réparties entre différentes zones AZs mais ne sont pas hébergées dans la même zone, une seule altération de la zone Z peut avoir un impact sur le processus sous-jacent end-to-end. Par exemple, si l'application A possède plusieurs répliques dans AZ 1 et AZ 2, mais que l'application B possède toutes ses répliques dans AZ 3, la perte de l'AZ 3 affectera tous les end-to-end processus entre ces deux charges de travail (application A et B). La combinaison des contraintes de dispersion topologique et de l'affinité des pods peut améliorer la résilience de votre application en répartissant

les pods sur l'ensemble AZs, ainsi qu'en configurant une relation entre certains pods pour garantir qu'ils sont colocalisés ensemble.

À l'aide des règles d'affinité des pods, vous pouvez définir des relations entre les charges de travail afin d'influencer le comportement du planificateur Kubernetes afin qu'il colocalise les pods sur le même nœud de travail ou dans la même AZ. Vous pouvez également configurer le niveau de rigueur de ces contraintes de planification.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products
  namespace: ecommerce
  labels:
    app.kubernetes.io/version: "0.1.6"

    spec:
      serviceAccountName: graphql-service-account
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - orders
            topologyKey: "kubernetes.io/hostname"
```

Le schéma ci-dessous décrit les pods qui ont été co-localisés sur le même nœud en utilisant des règles d'affinité des pods.

Vérifiez que votre environnement de cluster peut gérer la perte d'un AZ

Après avoir satisfait aux exigences ci-dessus, la prochaine étape importante consiste à vérifier que vous disposez d'une capacité de calcul et de charge de travail suffisante pour faire face à la perte d'un AZ. Vous pouvez le faire en déclenchant manuellement un changement de zone dans EKS. Vous pouvez également activer l'autoshift zonal et configurer des essais pratiques pour vérifier que vos applications fonctionnent comme prévu avec un AZ de moins dans votre environnement de cluster.

Questions fréquentes (FAQ)

Pourquoi utiliser cette fonctionnalité ?

En utilisant le décalage de zone ARC ou le décalage automatique de zone dans votre cluster EKS, vous pouvez mieux maintenir la disponibilité des applications Kubernetes en automatisant le processus de restauration rapide qui consiste à déplacer le trafic réseau interne au cluster vers

une zone AZ altérée. Grâce à l'ARC, vous pouvez éviter des étapes longues et compliquées qui entraînent souvent une période de récupération prolongée en cas d'altération de l'AZ.

Comment fonctionne cette fonctionnalité avec les autres AWS services ?

EKS s'intègre à ARC, qui fournit l'interface principale dans laquelle vous pouvez effectuer des opérations de restauration AWS. Pour garantir que le trafic interne au cluster est correctement acheminé hors d'une zone de disponibilité altérée, des modifications sont apportées à la liste des points de terminaison réseau pour les pods exécutés dans le plan de données Kubernetes. Si vous utilisez des équilibreurs de AWS charge pour acheminer le trafic externe vers le cluster, vous pouvez déjà enregistrer vos équilibreurs de charge auprès de l'ARC et déclencher un changement de zone pour empêcher le trafic de circuler dans la zone dégradée. Cette fonctionnalité interagit également avec les Amazon EC2 Auto Scaling Groups (ASG) créés par EKS Managed Node Groups (MNG). Pour empêcher qu'un AZ altéré ne soit utilisé pour de nouveaux pods Kubernetes ou pour le lancement de nouveaux nœuds, EKS supprime l'AZ altéré de l'ASG.

En quoi cette fonctionnalité diffère-t-elle des protections Kubernetes par défaut ?

Cette fonctionnalité fonctionne en tandem avec plusieurs protections intégrées natives de Kubernetes qui aident les clients à rester résilients. Vous pouvez configurer des sondes de disponibilité et de réactivité du pod qui déterminent à quel moment un pod doit recevoir du trafic. Lorsque ces sondes échouent, Kubernetes supprime ces pods en tant que cibles d'un service et le trafic n'est plus envoyé vers le pod. Bien que cela soit utile, il n'est pas trivial pour les clients de configurer ces contrôles de santé de manière à garantir leur échec en cas de dégradation d'une zone. La fonction de changement de zone ARC vous fournit un filet de sécurité supplémentaire qui les aide à isoler complètement une zone de zone dégradée lorsque les protections natives de Kubernetes ne sont pas suffisantes. Il vous permet également de tester facilement l'état de préparation opérationnelle et la résilience de votre architecture.

Puis-je AWS déclencher un changement de zone en mon nom ?

Oui, si vous souhaitez utiliser le décalage de zone ARC de manière entièrement automatisée, vous pouvez activer le décalage automatique de zone ARC. Avec l'autoshift zonal, vous pouvez compter sur vous AWS pour surveiller l'état de santé de votre cluster EKS et AZs pour déclencher automatiquement un changement de vitesse lorsqu'une altération de l'AZ est détectée.

Que se passe-t-il si j'utilise cette fonctionnalité et que mes nœuds de travail et mes charges de travail ne sont pas prédimensionnés ?

Si vous n'êtes pas prédimensionné et que vous comptez sur le provisionnement de nœuds ou de pods supplémentaires lors d'un changement de zone, vous risquez de connaître un retard de restauration. Le processus d'ajout de nouveaux nœuds au plan de données Kubernetes prendra un certain temps, ce qui peut avoir des répercussions sur les performances et la disponibilité en temps réel de vos applications, en particulier en cas de détérioration zonale. En outre, dans le cas d'une altération zonale, vous pouvez rencontrer une contrainte de capacité de calcul potentielle qui empêcherait l'ajout de nouveaux nœuds au nœud sain AZs.

Si vos charges de travail ne sont pas prédimensionnées et réparties sur l'ensemble AZs de votre cluster, une altération zonale peut avoir un impact sur la disponibilité d'une application qui s'exécute uniquement sur les nœuds de travail dans une zone de zone affectée. Afin d'atténuer le risque d'une interruption complète de la disponibilité de votre application, EKS dispose d'un dispositif de sécurité intégré permettant d'envoyer le trafic vers les points de terminaison du Pod situés dans une zone affectée si tous les points de terminaison de cette charge de travail se trouvent dans une zone de non-conformité. Cependant, il est vivement recommandé de pré-dimensionner et de répartir vos applications sur l'ensemble du site AZs afin de maintenir la disponibilité en cas de problème de zone.

Que se passe-t-il si j'exécute une application dynamique ?

Si vous exécutez une application dynamique, vous devez évaluer sa tolérance aux pannes en fonction du cas d'utilisation et de l'architecture. Si vous avez une architecture ou un modèle actif/en veille, il se peut que l'actif se trouve dans une zone de disponibilité altérée. Au niveau de l'application, si le mode veille n'est pas activé, vous pouvez rencontrer des problèmes avec votre application. Vous pouvez également rencontrer des problèmes lorsque de nouveaux pods Kubernetes sont lancés en bon état, AZs car ils ne pourront pas se connecter aux volumes persistants liés à l'AZ altérée.

Cette fonctionnalité fonctionne-t-elle avec Karpenter ?

Le support Karpenter n'est actuellement pas disponible avec le changement de zone ARC et le changement automatique de zone dans EKS. Si un AZ est altéré, vous pouvez ajuster la NodePool configuration Karpenter appropriée en supprimant l'AZ défectueux afin que les nouveaux nœuds de travail ne soient lancés que dans le nœud sain AZs.

Cette fonctionnalité fonctionne-t-elle avec EKS Fargate ?

Cette fonctionnalité ne fonctionne pas avec EKS Fargate. Par défaut, lorsqu'EKS Fargate reconnaît un événement sanitaire dans une zone, Pods préfère fonctionner dans l'autre zone. AZs

Le plan de contrôle Kubernetes géré par EKS sera-t-il impacté ?

Non, par défaut, Amazon EKS exécute et adapte le plan de contrôle Kubernetes sur plusieurs plans afin de garantir une AZs haute disponibilité. Le décalage de zone ARC et le décalage automatique de zone n'agiront que sur le plan de données Kubernetes.

Y a-t-il des coûts associés à cette nouvelle fonctionnalité ?

Vous pouvez utiliser le décalage de zone ARC et le décalage automatique de zone dans votre cluster EKS sans frais supplémentaires. Cependant, vous continuerez à payer pour les instances provisionnées et il est vivement recommandé de prédimensionner votre plan de données Kubernetes avant d'utiliser cette fonctionnalité. Vous devez trouver le juste équilibre entre le coût et la disponibilité des applications.

Ressources supplémentaires

- Comment fonctionne un changement de zone
- Meilleures pratiques pour les changements de zone dans l'ARC
- Ressources et scénarios pris en charge pour le changement de zone et le changement automatique de zone
- Gérer des charges de travail résilientes sur Amazon EKS
- Éliminez le retard de mise à l'échelle des nœuds Kubernetes grâce à la priorité des pods et au surprovisionnement
- Adaptez les pods CoreDNS à un trafic DNS élevé

Activez EKS Zonal Shift pour éviter les zones de disponibilité altérées

Amazon Application Recovery Controller (ARC) vous aide à gérer et à coordonner la restauration de vos applications dans toutes les zones de disponibilité (AZs) et fonctionne avec de nombreux services, dont Amazon EKS. Grâce à la prise en charge du changement de zone ARC par EKS, vous pouvez déplacer le trafic réseau interne au cluster vers une zone AZ altérée. Vous pouvez également autoriser AWS à surveiller l'état de votre AZ en mauvais état AZs et à transférer temporairement le trafic réseau hors de portée en votre nom.

Comment utiliser EKS Zonal Shift :

1. Activez votre cluster EKS avec Amazon Application Recovery Controller (ARC). Cela se fait au niveau du cluster à l'aide de la console Amazon EKS, de la AWS CLI ou de eksctl. CloudFormation

2. Une fois activé, vous pouvez gérer les changements de zone ou les changements automatiques de zone à l'aide de la console ARC, de la AWS CLI ou du décalage de zone et de changement automatique de zone. APIs

Notez qu'après avoir enregistré un cluster EKS auprès d'ARC, vous devez toujours configurer ARC. Par exemple, vous pouvez utiliser la console ARC pour configurer Zonal Autoshift.

Pour des informations plus détaillées sur le fonctionnement d'EKS Zonal Shift et sur la façon de concevoir vos charges de travail pour gérer les zones de disponibilité altérées, consultez. the section called "En savoir plus sur Zonal Shift"

Considérations

- Le mode automatique EKS ne prend pas en charge Amazon Application Recovery Controller, Zonal Shift et Zonal Autoshift.
- Nous recommandons d'attendre au moins 60 secondes entre les opérations de changement de zone afin de garantir le traitement approprié de chaque demande.

  Lorsque vous essayez d'effectuer des changements de zone en succession rapide (à 60 secondes d'intervalle), le service Amazon EKS risque de ne pas traiter correctement toutes les demandes de quart de travail. Cela est dû au mécanisme de sondage actuel qui met à jour l'état zonal du cluster. Si vous devez effectuer plusieurs changements de zone, assurez-vous qu'il y a suffisamment de temps entre les opérations pour que le système puisse traiter chaque modification.

Qu'est-ce qu'Amazon Application Recovery Controller ?

Amazon Application Recovery Controller (ARC) vous aide à préparer et à accélérer la restauration des applications qui s'exécutent dessus AWS. Le changement de zone vous permet de vous remettre rapidement en cas de défaillance d'une zone de disponibilité (AZ), en déplaçant temporairement le trafic d'une ressource prise en charge d'une zone Z vers un trafic normal AZs dans la AWS région.

En savoir plus sur Amazon Application Recovery Controller (ARC)

Qu'est-ce que le décalage zonal ?

Le changement de zone est une fonctionnalité d'ARC qui vous permet de déplacer le trafic vers une ressource telle qu'un cluster EKS ou un Elastic Load Balancer hors d'une zone de disponibilité d' AWS une région afin d'atténuer rapidement un problème et de récupérer rapidement

votre application. Vous pouvez choisir de transférer le trafic, par exemple, parce qu'un mauvais déploiement entraîne des problèmes de latence ou parce que la zone de disponibilité est altérée. Un changement de zone ne nécessite aucune étape de configuration préalable.

En savoir plus sur ARC Zonal Shift

Qu'est-ce que l'autoshift zonal ?

L'autoshift zonal est une fonctionnalité d'ARC que vous pouvez activer pour autoriser AWS le transfert du trafic d'un AZ pour les ressources prises en charge, en votre nom, vers un trafic sain AZs dans la AWS région. AWS lance un changement automatique lorsque la télémétrie interne indique la présence d'une anomalie dans un AZ d'une région susceptible d'avoir un impact sur les clients. La télémétrie interne intègre des métriques provenant de plusieurs sources, notamment le AWS réseau et les services Amazon EC2 et Elastic Load Balancing.

AWS met fin aux changements automatiques lorsque les indicateurs indiquent qu'il n'y a plus de problème ou de problème potentiel.

En savoir plus sur ARC Zonal Autoshift

Que fait EKS lors d'un changement automatique ?

EKS met à jour les configurations réseau afin d'éviter de rediriger le trafic vers des destinations altérées AZs. De plus, si vous utilisez des groupes de nœuds gérés, EKS ne lancera de nouveaux nœuds dans les nœuds sains que AZs lors d'un changement de zone. Lorsque le quart de travail expire ou est annulé, les configurations réseau sont restaurées pour inclure l'AZ précédemment détecté comme défectueux.

En savoir plus sur EKS Zonal Shift.

Enregistrer le cluster EKS auprès d'Amazon Application Recovery Controller (ARC) (AWS console)

1. Trouvez le nom et la région du cluster EKS que vous souhaitez enregistrer auprès d'ARC.
2. Accédez à la console EKS de cette région, puis sélectionnez votre cluster.
3. Sur la page d'informations du cluster, sélectionnez l'onglet Vue d'ensemble.
4. Sous le titre Zonal Shift, sélectionnez le bouton Gérer.
5. Sélectionnez activer ou désactiver pour EKS Zonal Shift.

Votre cluster EKS est maintenant enregistré auprès d'ARC.

Si vous AWS souhaitez détecter et éviter les zones de disponibilité altérées, vous devez configurer ARC Zonal Autoshift. Par exemple, vous pouvez le faire dans la console ARC.

Étapes suivantes

- Découvrez comment activer le changement automatique par zone.
- Découvrez comment démarrer manuellement un changement de zone.

Découvrez comment fonctionne le contrôle d'accès dans Amazon EKS

Découvrez comment gérer l'accès à votre cluster Amazon EKS. L'utilisation d'Amazon EKS nécessite de connaître la manière dont Kubernetes et AWS Identity and Access Management (AWS IAM) gèrent le contrôle d'accès.

Cette section inclut :

the section called "Accès à l'API Kubernetes"— Découvrez comment permettre aux applications ou aux utilisateurs de s'authentifier auprès de l'API Kubernetes. Vous pouvez utiliser des entrées d'accès, l'aws-auth ou un ConfigMap fournisseur OIDC externe.

the section called "Accès aux ressources du cluster"— Découvrez comment configurer le AWS Management Console pour communiquer avec votre cluster Amazon EKS. Utilisez la console pour afficher les ressources Kubernetes du cluster, telles que les espaces de noms, les nœuds et les pods.

the section called "Cluster d'accès avec kubectl"— Découvrez comment configurer kubectl pour communiquer avec votre cluster Amazon EKS. Utilisez la AWS CLI pour créer un fichier kubeconfig.

the section called "Accès à la charge de travail à AWS "— Découvrez comment associer un compte de service Kubernetes à IAM Roles AWS . Vous pouvez utiliser Pod Identity ou IAM Roles for Service Accounts (IRSA).

Tâches communes

- Accordez aux développeurs l'accès à l'API Kubernetes. Consultez les ressources Kubernetes dans le. AWS Management Console
  - Solution : utilisez des entrées d'accès pour associer les autorisations Kubernetes RBAC aux utilisateurs ou aux rôles AWS IAM.
- Configurez kubectl pour communiquer avec un cluster Amazon EKS à l'aide AWS des informations d'identification.
  - Solution : utilisez la AWS CLI pour créer un fichier kubeconfig.
- Utilisez un fournisseur d'identité externe, tel que Ping Identity, pour authentifier les utilisateurs auprès de l'API Kubernetes.

- Solution : associez un fournisseur OIDC externe.
- Donnez aux charges de travail de votre cluster Kubernetes la possibilité d'appeler. AWS APIs
  - Solution : utilisez Pod Identity pour associer un rôle AWS IAM à un compte de service Kubernetes.

Contexte

- Découvrez comment fonctionnent les comptes de service Kubernetes.
- Passez en revue le modèle de contrôle d'accès basé sur les rôles (RBAC) de Kubernetes
- Pour plus d'informations sur la gestion de l'accès aux AWS ressources, consultez le guide de l'utilisateur AWS IAM. Vous pouvez également suivre une formation d'introduction gratuite sur l'utilisation d' AWS IAM.

Considérations relatives au mode automatique EKS

Le mode automatique EKS s'intègre aux entrées d'accès EKS Pod Identity et EKS EKS.

- Le mode automatique EKS utilise des entrées d'accès pour accorder des autorisations Kubernetes au plan de contrôle EKS. Par exemple, les politiques d'accès permettent au mode automatique d'EKS de lire des informations sur les points de terminaison et les services du réseau.
  - Vous ne pouvez pas désactiver les entrées d'accès sur un cluster en mode automatique EKS.
  - Vous pouvez éventuellement activer le `aws-authConfigMap`.
  - Les entrées d'accès pour le mode automatique EKS sont automatiquement configurées. Vous pouvez consulter ces entrées d'accès, mais vous ne pouvez pas les modifier.
  - Si vous utilisez un NodeClass pour créer un rôle Node IAM personnalisé, vous devez créer une entrée d'accès pour le rôle conformément à la politique EKSAuto NodePolicy d'accès Amazon.
- Si vous souhaitez accorder des autorisations aux charges de travail pour les AWS services, utilisez EKS Pod Identity.
  - Il n'est pas nécessaire d'installer l'agent Pod Identity sur les clusters EKS Auto Mode.

Accorder aux utilisateurs et aux rôles IAM l'accès à Kubernetes APIs

Votre cluster possède un point de terminaison d'API Kubernetes. Kubectl utilise cette API. Vous pouvez vous authentifier auprès de cette API à l'aide de deux types d'identités :

- Un responsable de la gestion des AWS identités et des accès (IAM) (rôle ou utilisateur) : ce type nécessite une authentification auprès d'IAM. Les utilisateurs peuvent se connecter en AWS tant qu'utilisateur IAM ou avec une identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. Les utilisateurs ne peuvent se connecter qu'avec une identité fédérée si votre administrateur a précédemment configuré une fédération d'identité avec des rôles IAM. Lorsque les utilisateurs accèdent AWS en utilisant la fédération, ils assument indirectement un rôle. Lorsque les utilisateurs utilisent ce type d'identité :
  - Vous pouvez leur attribuer des autorisations Kubernetes afin qu'ils puissent travailler avec des objets Kubernetes sur votre cluster. Pour plus d'informations sur la façon d'attribuer des autorisations à vos principaux IAM afin qu'ils puissent accéder aux objets Kubernetes de votre cluster, consultez. the section called “Accorder des autorisations”
  - Vous pouvez leur attribuer des autorisations IAM afin qu'ils puissent travailler avec votre cluster Amazon EKS et ses ressources à l'aide de l'API, de la AWS CLI AWS CloudFormation AWS Management Console, ou `eksctl` d'Amazon EKS. Pour plus d'informations, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service.
  - Les nœuds rejoignent votre cluster en endossant un rôle IAM. La possibilité d'accéder à votre cluster à l'aide des principes IAM est fournie par l'authentificateur AWS IAM pour Kubernetes, qui s'exécute sur le plan de contrôle Amazon EKS.
- Un utilisateur de votre propre fournisseur OpenID Connect (OIDC) : ce type nécessite une authentification auprès de votre fournisseur OIDC. Pour plus d'informations sur la configuration de votre propre fournisseur OIDC avec votre cluster Amazon EKS, consultezthe section called “Lien vers le fournisseur OIDC”. Lorsque les utilisateurs utilisent ce type d'identité :
  - Vous pouvez leur attribuer des autorisations Kubernetes afin qu'ils puissent travailler avec des objets Kubernetes sur votre cluster.
  - Impossible de leur attribuer des autorisations IAM afin qu'ils puissent travailler avec votre cluster Amazon EKS et ses ressources à l'aide de l'API, de la AWS CLI ou `eksctl` de Amazon EKS. AWS CloudFormation AWS Management Console

Vous pouvez utiliser les deux types d'identités avec votre cluster. La méthode d'authentification IAM ne peut pas être désactivée. La méthode d'authentification OIDC est facultative.

Associer les identités IAM aux autorisations Kubernetes

L'authentificateur AWS IAM pour Kubernetes est installé sur le plan de contrôle de votre cluster. Il permet aux principes (rôles et utilisateurs) d'AWS Identity and Access Management (IAM) que vous autorisez à accéder aux ressources Kubernetes de votre cluster. Vous pouvez autoriser les principaux IAM à accéder aux objets Kubernetes de votre cluster en utilisant l'une des méthodes suivantes :

- Création d'entrées d'accès : si votre cluster correspond à la version de plate-forme répertoriée dans la section Conditions requises pour la version Kubernetes de votre cluster ou est ultérieure à celle-ci, nous vous recommandons d'utiliser cette option.

  Utilisez les entrées d'accès pour gérer les autorisations Kubernetes des principaux IAM depuis l'extérieur du cluster. Vous pouvez ajouter et gérer l'accès au cluster à l'aide de l'API EKS, de l'interface de ligne de AWS commande AWS SDKs AWS CloudFormation,, et AWS Management Console. Cela signifie que vous pouvez gérer les utilisateurs avec les mêmes outils que ceux avec lesquels vous avez créé le cluster.

  Pour commencer, suivez eks/latest/userguide/setting - up-access-entries .html [Changez le mode d'authentification pour utiliser les entrées d'accès, type="documentation "], puis migrer les entrées ConfigMap aws-auth existantes pour accéder aux entrées.
- Ajouter des entrées au **`aws-auth ConfigMap`** — Si la version de la plate-forme de votre cluster est antérieure à la version répertoriée dans la section Conditions préalables, vous devez utiliser cette option. Si la version de plate-forme de votre cluster est identique ou ultérieure à la version de plate-forme répertoriée dans la section Conditions requises pour la version Kubernetes de votre cluster, et que vous y avez ajouté des entrées`ConfigMap`, nous vous recommandons de migrer ces entrées vers des entrées d'accès. Vous ne pouvez `ConfigMap` toutefois pas migrer les entrées qu'Amazon EKS y a ajoutées, telles que les entrées pour les rôles IAM utilisés avec des groupes de nœuds gérés ou des profils Fargate. Pour de plus amples informations, veuillez consulter the section called “Accès à l'API Kubernetes”.
  - Si vous devez utiliser l'option `ConfigMap aws-auth`, vous pouvez ajouter des entrées à la `ConfigMap` à l'aide de la commande `eksctl create iamidentitymapping`. Pour plus d'informations, consultez la section Gérer les utilisateurs et les rôles IAM dans la documentation `eksctl`.

Définir le mode d'authentification du cluster

Chaque cluster possède un mode d'authentification. Le mode d'authentification détermine les méthodes que vous pouvez utiliser pour autoriser les principaux IAM à accéder aux objets Kubernetes de votre cluster. Il existe trois modes d'authentification.

> **Important**
>
> Une fois que la méthode de saisie d'accès est activée, elle ne peut pas être désactivée.
> Si la `ConfigMap` méthode n'est pas activée lors de la création du cluster, elle ne pourra pas être activée ultérieurement. La `ConfigMap` méthode est activée pour tous les clusters créés avant l'introduction des entrées d'accès.
> Si vous utilisez des nœuds hybrides avec votre cluster, vous devez utiliser les modes d'authentification `API` ou de `API_AND_CONFIG_MAP` cluster.

L'**`aws-authConfigMap`**intérieur du cluster

Il s'agit du mode d'authentification d'origine pour les clusters Amazon EKS. Le principal IAM qui a créé le cluster est l'utilisateur initial à avoir accès au cluster au moyen de `kubectl`. L'utilisateur initial doit ajouter d'autres utilisateurs à la liste dans la `ConfigMap` `aws-auth` et attribuer des autorisations qui affectent les autres utilisateurs au sein du cluster. Ces autres utilisateurs ne peuvent pas gérer ou supprimer l'utilisateur initial, car il n'y a aucune entrée `ConfigMap` à gérer dans le.

À la fois les entrées **`ConfigMap`** et les entrées d'accès

Avec ce mode d'authentification, vous pouvez utiliser les deux méthodes pour ajouter des principaux IAM au cluster. Notez que chaque méthode stocke des entrées distinctes ; par exemple, si vous ajoutez une entrée d'accès depuis la AWS CLI, elle n'`aws-authConfigMap`est pas mise à jour.

Accès aux entrées uniquement

Avec ce mode d'authentification, vous pouvez utiliser l'API EKS AWS SDKs AWS CloudFormation, l'interface de ligne de AWS commande et AWS Management Console gérer l'accès au cluster pour les principaux IAM.

Chaque entrée d'accès possède un type et vous pouvez utiliser la combinaison d'une portée d'accès pour limiter le principal à un espace de noms spécifique et d'une stratégie d'accès pour définir des politiques d'autorisations réutilisables préconfigurées. Vous pouvez également

utiliser le type STANDARD et les groupes Kubernetes RBAC pour attribuer des autorisations personnalisées.

Mode d'authentification	Méthodes
`ConfigMap` uniquement (`CONFIG_MAP`)	`aws-auth ConfigMap`
API EKS et `ConfigMap` (`API_AND_CONFIG_MAP`)	accéder aux entrées dans l'API EKS, l'interface de ligne de AWS commande AWS SDKs AWS CloudFormation,, AWS Management Console et `aws-auth ConfigMap`
API EKS uniquement (`API`)	accéder aux entrées de l'API EKS, de l'interface de ligne de AWS commande AWS SDKs AWS CloudFormation, et AWS Management Console

> **Note**
> Le mode automatique d'Amazon EKS nécessite des entrées Access.

Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS

Cette section est conçue pour vous montrer comment gérer l'accès principal IAM aux clusters Kubernetes dans Amazon Elastic Kubernetes Service (EKS) à l'aide d'entrées et de politiques d'accès. Vous y trouverez des informations sur la modification des modes d'authentification, la migration depuis des `aws-auth` ConfigMap entrées existantes, la création, la mise à jour et la suppression d'entrées d'accès, l'association de politiques aux entrées, la révision des autorisations de politique prédéfinies, ainsi que les principales conditions et considérations relatives à la gestion sécurisée des accès.

Présentation

Les entrées d'accès EKS constituent le meilleur moyen d'autoriser les utilisateurs à accéder à l'API Kubernetes. Par exemple, vous pouvez utiliser des entrées d'accès pour autoriser les développeurs

à utiliser kubectl. Fondamentalement, une entrée d'accès EKS associe un ensemble d'autorisations Kubernetes à une identité IAM, telle qu'un rôle IAM. Par exemple, un développeur peut assumer un rôle IAM et l'utiliser pour s'authentifier auprès d'un cluster EKS.

Fonctionnalités

- Authentification et autorisation centralisées : contrôle l'accès aux clusters Kubernetes directement via Amazon EKS APIs, éliminant ainsi le besoin de basculer entre Kubernetes AWS et APIs Kubernetes pour obtenir les autorisations des utilisateurs.
- Gestion granulaire des autorisations : utilise des entrées et des politiques d'accès pour définir des autorisations précises pour les principaux AWS IAM, notamment en modifiant ou en révoquant l'accès du créateur à l'administrateur du cluster.
- Intégration des outils IaC : prend en charge l'infrastructure sous forme d'outils de code tels AWS CloudFormation que Terraform et AWS CDK pour définir les configurations d'accès lors de la création du cluster.
- Restauration après une mauvaise configuration : permet de restaurer l'accès au cluster via l'API Amazon EKS sans accès direct à l'API Kubernetes.
- Réduction des frais généraux et sécurité améliorée : centralise les opérations pour réduire les frais tout en tirant parti des fonctionnalités AWS IAM telles que la journalisation des CloudTrail audits et l'authentification multifactorielle.

Comment associer des autorisations

Vous pouvez associer des autorisations Kubernetes pour accéder aux entrées de deux manières :

- Utilisez une politique d'accès. Les politiques d'accès sont des modèles d'autorisations Kubernetes prédéfinis gérés par. AWS Pour de plus amples informations, veuillez consulter the section called "Passez en revue les politiques d'accès".
- Référencez un groupe Kubernetes. Si vous associez une identité IAM à un groupe Kubernetes, vous pouvez créer des ressources Kubernetes qui accordent des autorisations au groupe. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Considérations

Lorsque vous activez les entrées d'accès EKS sur des clusters existants, gardez à l'esprit les points suivants :

- Comportement des clusters existants : pour les clusters créés avant l'introduction des entrées d'accès (ceux dont les versions initiales de la plate-forme sont antérieures à celles spécifiées dans les exigences relatives aux versions de la plate-forme), EKS crée automatiquement une entrée d'accès reflétant les autorisations préexistantes. Cette entrée inclut l'identité IAM qui a initialement créé le cluster et les autorisations administratives accordées à cette identité lors de la création du cluster.
- Gestion de l'héritage **`aws-auth`** ConfigMap : si votre cluster s'appuie sur l'héritage `aws-auth` ConfigMap pour la gestion des accès, seule l'entrée d'accès du créateur du cluster d'origine est automatiquement créée lors de l'activation des entrées d'accès. Les rôles ou autorisations supplémentaires ajoutés au ConfigMap (par exemple, les rôles IAM personnalisés pour les développeurs ou les services) ne sont pas automatiquement migrés. Pour résoudre ce problème, créez manuellement les entrées d'accès correspondantes.

Mise en route

1. Déterminez la politique d'identité et d'accès IAM que vous souhaitez utiliser.
   - the section called "Passez en revue les politiques d'accès"
2. Activez les entrées d'accès EKS sur votre cluster. Vérifiez que vous disposez d'une version de plateforme prise en charge.
   - the section called "Mode d'authentification"
3. Créez une entrée d'accès qui associe une identité IAM à une autorisation Kubernetes.
   - the section called "Création d'entrées d'accès"
4. Authentifiez-vous auprès du cluster à l'aide de l'identité IAM.
   - the section called "Configuration de la AWS CLI"
   - the section called "Configurer kubectl et eksctl"

Associer les politiques d'accès aux entrées d'accès

Vous pouvez attribuer une ou plusieurs stratégies d'accès à des entrées d'accès de type `STANDARD`. Amazon EKS accorde automatiquement aux autres types d'entrées d'accès les autorisations nécessaires pour fonctionner correctement dans votre cluster. Les politiques d'accès d'Amazon EKS incluent les autorisations Kubernetes, et non les autorisations IAM. Avant d'associer une politique d'accès à une entrée d'accès, assurez-vous de connaître les autorisations Kubernetes incluses dans chaque politique d'accès. Pour de plus amples informations, veuillez consulter the section

called "Passez en revue les politiques d'accès". Si aucune des politiques d'accès ne répond à vos exigences, n'associez aucune politique d'accès à une entrée d'accès. Spécifiez plutôt un ou plusieurs noms de groupe pour l'entrée d'accès et créez et gérez les objets de contrôle d'accès basés sur les rôles Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Création d'entrées d'accès".

- Une entrée d'accès existante. Pour en créer un, consultez the section called "Création d'entrées d'accès".
- Un rôle ou un utilisateur AWS Identity and Access Management disposant des autorisations suivantes : `ListAccessEntriesDescribeAccessEntry`,`UpdateAccessEntry`,`ListAccessPolicies`,`Associ` et`DisassociateAccessPolicy`. Pour plus d'informations, consultez la section Actions définies par Amazon Elastic Kubernetes Service dans le Service Authorization Reference.

Avant d'associer des stratégies d'accès à des entrées d'accès, tenez compte des exigences suivantes :

- Vous pouvez associer plusieurs stratégies d'accès à chaque entrée d'accès, mais vous ne pouvez associer chaque politique à une entrée d'accès qu'une seule fois. Si vous associez plusieurs politiques d'accès, le principal IAM de l'entrée d'accès dispose de toutes les autorisations incluses dans toutes les politiques d'accès associées.
- Vous pouvez définir une politique d'accès à toutes les ressources d'un cluster ou en spécifiant le nom d'un ou de plusieurs espaces de noms Kubernetes. Vous pouvez utiliser des caractères génériques pour le nom d'un espace de noms. Par exemple, si vous souhaitez étendre une stratégie d'accès à tous les espaces de noms commençant par `dev-`, vous pouvez spécifier `dev-*` sous forme de nom d'espace de noms. Assurez-vous que les espaces de noms existent sur votre cluster et que votre orthographe correspond au nom réel de l'espace de noms du cluster. Amazon EKS ne confirme ni l'orthographe ni l'existence des espaces de noms de votre cluster.
- Vous pouvez modifier la portée d'accès d'une stratégie d'accès après l'avoir associée à une entrée d'accès. Si vous avez défini la politique d'accès aux espaces de noms Kubernetes, vous pouvez ajouter et supprimer des espaces de noms pour l'association, si nécessaire.
- Si vous associez une stratégie d'accès à une entrée d'accès dont les noms de groupe sont également spécifiés, le principal IAM dispose alors de toutes les autorisations dans toutes les stratégies d'accès associées. Il dispose également de toutes les autorisations dans n'importe quel Kubernetes `Role` ou `ClusterRole` objet spécifié dans n'importe quel Kubernetes `Role` et d'`RoleBinding`objets spécifiant les noms de groupes.

- Si vous exécutez la `kubectl auth can-i --list` commande, vous ne verrez aucune autorisation Kubernetes attribuée par les politiques d'accès associées à une entrée d'accès pour le principal IAM que vous utilisez lorsque vous exécutez la commande. La commande affiche uniquement les autorisations Kubernetes si vous les avez accordées dans Kubernetes `Role` ou dans des `ClusterRole` objets que vous avez liés aux noms de groupe ou au nom d'utilisateur que vous avez spécifiés pour une entrée d'accès.
- Si vous vous faites passer pour un utilisateur ou un groupe Kubernetes lorsque vous interagissez avec des objets Kubernetes sur votre cluster, par exemple en utilisant la `kubectl` commande avec `--as` *`username`* ou`--as-group` *`group-name`* , vous forcez l'utilisation de l'autorisation Kubernetes RBAC. Par conséquent, le principal IAM ne dispose d'aucune autorisation attribuée par les stratégies d'accès associées à l'entrée d'accès. Les seules autorisations Kubernetes dont dispose l'utilisateur ou le groupe dont le principal IAM se fait passer sont les autorisations Kubernetes que vous lui avez accordées dans Kubernetes `Role` ou les `ClusterRole` objets que vous avez liés aux noms de groupes ou aux noms d'utilisateur. Pour que votre principal IAM dispose des autorisations définies dans les politiques d'accès associées, ne vous faites pas passer pour un utilisateur ou un groupe Kubernetes. Le principal IAM aura également toutes les autorisations que vous lui avez accordées dans les Kubernetes `Role` ou les `ClusterRole` objets que vous avez liés aux noms de groupe ou au nom d'utilisateur que vous avez spécifiés pour l'entrée d'accès. Pour plus d'informations, consultez la section Usurpation d'identité d'un utilisateur dans la documentation de Kubernetes.

Vous pouvez associer une politique d'accès à une entrée d'accès à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster qui possède une entrée d'accès à laquelle associer une stratégie d'accès.
3. Choisissez l'onglet Access.
4. Si le type d'entrée d'accès est Standard, vous pouvez associer ou dissocier les stratégies d'accès Amazon EKS. Si le type de votre entrée d'accès est autre que Standard, cette option n'est pas disponible.
5. Choisissez Associer la stratégie d'accès.

6. Dans Nom de la stratégie, sélectionnez la stratégie avec les autorisations que vous souhaitez attribuer au principal IAM. Pour consulter les autorisations incluses dans chaque stratégie, consultez the section called "Passez en revue les politiques d'accès".
7. Pour Portée d'accès, choisissez une portée d'accès. Si vous choisissez Cluster, les autorisations définies dans la politique d'accès sont accordées au principal IAM pour les ressources de tous les espaces de noms Kubernetes. Si vous choisissez l'espace de noms Kubernetes, vous pouvez ensuite choisir Ajouter un nouvel espace de noms. Dans le champ Namespace qui apparaît, vous pouvez saisir le nom d'un espace de noms Kubernetes sur votre cluster. Si vous souhaitez que le principal IAM dispose d'autorisations sur plusieurs espaces de noms, vous pouvez saisir plusieurs espaces de noms.
8. Choisissez Ajouter une stratégie d'accès.

AWS CLI

1. Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
2. Affichez les stratégies d'accès disponibles.

```
aws eks list-access-policies --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------------------------------------------------------------
|                                          ListAccessPolicies
                  |
+---------------------------------------------------------------------------------------------
+
||                                           accessPolicies
                  ||
```

```
|+-------------------------------------------------------------------
+------------------------------+|
||                                arn                                |
 name              ||
|+-------------------------------------------------------------------
+------------------------------+|
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSAdminPolicy       |
 AmazonEKSAdminPolicy         ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSEditPolicy        |
 AmazonEKSEditPolicy          ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSViewPolicy        |
 AmazonEKSViewPolicy          ||
|+-------------------------------------------------------------------
+------------------------------+|
```

Pour consulter les autorisations incluses dans chaque stratégie, consultez the section called "Passez en revue les politiques d'accès".

3. Consultez vos entrées d'accès existantes. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks list-access-entries --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.

```
{
    "accessEntries": [
        "arn:aws: iam::111122223333:role/my-role",
        "arn:aws: iam::111122223333:user/my-user"
    ]
}
```

4. Associez une stratégie d'accès à une entrée d'accès. L'exemple suivant associe la stratégie d'accès `AmazonEKSViewPolicy` à une entrée d'accès. Chaque fois que le rôle *my-role* IAM tente d'accéder à des objets Kubernetes sur le cluster, Amazon EKS autorise le rôle à utiliser les autorisations définies dans la politique pour accéder aux objets Kubernetes dans les espaces de noms Kubernetes et Kubernetes uniquement. *my-namespace1* *my-namespace2* *my-cluster*Remplacez-le par le nom de votre cluster, *111122223333* par votre identifiant de AWS compte et *my-role* par le nom du rôle IAM pour lequel vous souhaitez qu'Amazon EKS autorise l'accès aux objets du cluster Kubernetes.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

Si vous souhaitez que le principal IAM dispose des autorisations à l'échelle du cluster, remplacez `type=namespace,namespaces=`*`my-namespace1`*`,`*`my-namespace2`* par `type=cluster`. Si vous souhaitez associer plusieurs stratégies d'accès à l'entrée d'accès, exécutez la commande plusieurs fois, chaque fois avec une stratégie d'accès unique. Chaque stratégie d'accès associée possède sa propre portée.

> **Note**
>
> Si vous souhaitez ultérieurement modifier la portée d'une stratégie d'accès associée, réexécutez la commande précédente avec la nouvelle portée. Par exemple, si vous souhaitez supprimer*my-namespace2*, vous devez exécuter à nouveau la commande en utilisant `type=namespace,namespaces=`*`my-namespace1`* uniquement. Si vous souhaitez modifier la portée de `namespace` à`cluster`, vous devez exécuter à nouveau la commande en utilisant`type=cluster`, en supprimant`type=namespace,namespaces=`*`my-namespace1`*`,`*`my-namespace2`* .

5. Déterminez les stratégies d'accès associées à une entrée d'accès.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn
 arn:aws: iam::111122223333:role/my-role
```

L'exemple qui suit illustre un résultat.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws: iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
```

```
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

Dans l'exemple précédent, le principal IAM pour cette entrée d'accès dispose d'autorisations d'affichage sur tous les espaces de noms du cluster et d'autorisations d'administrateur pour deux espaces de noms Kubernetes.

6. Dissociez une stratégie d'accès d'une entrée d'accès. Dans cet exemple, la stratégie `AmazonEKSAdminPolicy` est dissociée d'une entrée d'accès. Le principal IAM conserve toutefois les autorisations définies dans la politique `AmazonEKSViewPolicy` d'accès pour les objets des *`my-namespace2`* espaces de noms *`my-namespace1`* et, car cette politique d'accès n'est pas dissociée de l'entrée d'accès.

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

Pour répertorier les politiques d'accès disponibles, voirthe section called “Passez en revue les politiques d'accès”.

Migration des entrées **`aws-auth ConfigMap`** existantes pour accéder aux entrées

Si vous avez ajouté des entrées `aws-auth ConfigMap` à votre cluster, nous vous recommandons de créer des entrées d'accès pour les entrées existantes de votre cluster `aws-authConfigMap`.

Après avoir créé les entrées d'accès, vous pouvez les supprimer de votre `ConfigMap`. Vous ne pouvez pas associer de politiques d'accès aux entrées du `aws-authConfigMap`. Si vous souhaitez associer des stratégies d'accès à vos principaux IAM, créez des entrées d'accès.

> **Important**
>
> - Lorsqu'un cluster est en mode `API_AND_CONFIGMAP` authentification et qu'il existe un mappage pour le même rôle IAM dans les entrées d'accès `aws-auth ConfigMap` et dans les entrées d'accès, le rôle utilise le mappage de l'entrée d'accès pour l'authentification. Les entrées d'accès ont priorité sur `ConfigMap` les entrées pour le même principal IAM.
> - Avant de supprimer des `aws-auth ConfigMap` entrées existantes créées par Amazon EKS pour un groupe de nœuds gérés ou un profil Fargate dans votre cluster, vérifiez si les entrées d'accès correctes pour ces ressources spécifiques existent dans votre cluster Amazon EKS. Si vous supprimez des entrées créées par Amazon EKS dans le `ConfigMap` sans disposer des entrées d'accès équivalentes, votre cluster ne fonctionnera pas correctement.

Prérequis

- Avoir des connaissances sur les entrées d'accès et les stratégies d'accès. Pour plus d'informations, consultez the section called "Accorder des autorisations" et the section called "Politiques d'accès associées".
- Un cluster existant avec une version de plate-forme égale ou ultérieure aux versions répertoriées dans les conditions préalables de la the section called "Accorder des autorisations" rubrique.
- Version `0.210.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
- Autorisations Kubernetes pour modifier le `aws-auth ConfigMap` dans l'espace de noms. `kube-system`
- Un rôle ou un utilisateur AWS Identity and Access Management disposant des autorisations suivantes : `CreateAccessEntry` et`ListAccessEntries`. Pour plus d'informations, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service.

eksctl

1. Consultez les entrées existantes dans votre `aws-auth ConfigMap`. Remplacez *my-cluster* par le nom de votre cluster.

```
eksctl get iamidentitymapping --cluster my-cluster
```

L'exemple qui suit illustre un résultat.

```
ARN
           USERNAME                                GROUPS
                       ACCOUNT
arn:aws: iam::111122223333:role/EKS-my-cluster-Admins
            Admins                                  system:masters
arn:aws: iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
            my-namespace-Viewers                    Viewers
arn:aws: iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
arn:aws: iam::111122223333:user/my-user
            my-user
arn:aws: iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws: iam::111122223333:role/EKS-my-cluster-managed-ng
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
```

2. the section called "Création d'entrées d'accès"pour toutes les `ConfigMap` entrées que vous avez créées renvoyées dans la sortie précédente. Lorsque vous créez les entrées d'accès, assurez-vous de spécifier les mêmes valeurs pour `ARN`, `USERNAME`, `GROUPS` et `ACCOUNT` et de les renvoyer dans votre résultat. Dans l'exemple de résultat, vous devez créer des entrées d'accès pour toutes les entrées sauf les deux dernières, étant donné que ces entrées ont été créées par Amazon EKS pour un profil Fargate et un groupe de nœuds géré.

3. Supprimez les entrées de la `ConfigMap` pour toutes les entrées d'accès que vous avez créées. Si vous ne supprimez pas l'entrée du`ConfigMap`, les paramètres de l'entrée d'accès pour l'ARN principal IAM remplacent l'`ConfigMap`entrée. *111122223333*Remplacez-le par votre numéro de AWS compte et *EKS-my-cluster-my-namespace-Viewers* par le nom du rôle dans l'entrée de votre`ConfigMap`. Si l'entrée que vous supprimez concerne un utilisateur IAM plutôt qu'un rôle IAM, remplacez-la par `user` et `role` *EKS-my-cluster-my-namespace-Viewers* par le nom d'utilisateur.

```
eksctl delete iamidentitymapping --arn arn:aws: iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

Vérifiez les autorisations relatives à la politique d'accès

Les politiques d'accès incluent `rules` celles qui contiennent Kubernetes `verbs` (autorisations) et. `resources` Les politiques d'accès n'incluent pas les autorisations ou les ressources IAM. Comme pour Kubernetes `Role` et les `ClusterRole` objets, les politiques d'accès incluent uniquement. `allow` `rules` Vous ne pouvez pas modifier le contenu d'une politique d'accès. Vous ne pouvez pas créer vos propres politiques d'accès. Si les autorisations définies dans les politiques d'accès ne répondent pas à vos besoins, créez des objets Kubernetes RBAC et spécifiez les noms de groupe pour vos entrées d'accès. Pour de plus amples informations, veuillez consulter the section called "Création d'entrées d'accès". Les autorisations contenues dans les politiques d'accès sont similaires à celles des rôles de cluster Kubernetes destinés aux utilisateurs. Pour plus d'informations, consultez la section Rôles destinés aux utilisateurs dans la documentation de Kubernetes.

Répertorier toutes les politiques

Utilisez l'une des politiques d'accès répertoriées sur cette page ou récupérez une liste de toutes les politiques d'accès disponibles à l'aide de la AWS CLI :

```
aws eks list-access-policies
```

Le résultat attendu doit ressembler à ceci (abrégé par souci de concision) :

```
{
    "accessPolicies": [
        {
            "name": "AmazonAIOpsAssistantPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonAIOpsAssistantPolicy"
        },
        {
            "name": "AmazonARCRegionSwitchScalingPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/
AmazonARCRegionSwitchScalingPolicy"
        },
        {
            "name": "AmazonEKSAdminPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy"
```

```
        },
        {
            "name": "AmazonEKSAdminViewPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy"
        },
        {
            "name": "AmazonEKSAutoNodePolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy"
        }
        // Additional policies omitted
    ]
}
```

EKSAdminPolitique d'Amazon

Cette stratégie d'accès inclut des autorisations qui accordent à un principal IAM le plus grand nombre d'autorisations sur les ressources. Lorsqu'elle est associée à une entrée d'accès, son étendue d'accès est généralement un ou plusieurs espaces de noms Kubernetes. Si vous souhaitez qu'un principal IAM dispose d'un accès administrateur à toutes les ressources de votre cluster, associez plutôt la stratégie d'accès the section called "Amazon EKSCluster AdminPolicy" à votre entrée d'accès.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
`apps`	`daemonsets` , `deployments` , `deployments/rollback` , `deployments/scale` , `replicasets` , `replicasets/scale`, `statefulsets` , `statefulsets/scale`	`create`, `delete`, `deletecollection` , `patch`, `update`
`apps`	`controllerrevisions` , `daemonsets` , `daemonsets/status` , `deployments` , `deployments/scale` , `deployments/status` ,	`get`, `list`, `watch`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
	replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
	`bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status`	`get`, `list`, `watch`
	`namespaces`	`get`,`list`, `watch`

Amazon EKSCluster AdminPolicy

Cette stratégie d'accès inclut des autorisations qui accordent à un administrateur principal IAM l'accès à un cluster. Lorsqu'elle est associée à une entrée d'accès, son étendue d'accès est généralement le cluster, plutôt qu'un espace de noms Kubernetes. Si vous souhaitez qu'un principal IAM ait une portée administrative plus limitée, pensez plutôt à associer la stratégie d'accès the section called "EKSAdminPolitique d'Amazon" à votre entrée d'accès.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

Groupes d'API Kubernetes	Kubernetes Non-Resource URLs	Ressources Kubernetes	Verbes Kubernetes (autorisations)
*		*	*
	*		*

Amazon EKSAdmin ViewPolicy

Cette politique d'accès inclut des autorisations qui accordent à un IAM principal l'accès à list/view toutes les ressources d'un cluster. Notez que cela inclut les secrets Kubernetes.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
*	*	get, list, watch

EKSEditPolitique d'Amazon

Cette politique d'accès inclut des autorisations qui permettent à un principal IAM de modifier la plupart des ressources Kubernetes.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizonta	get, list, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
	lpodautoscalers/status	
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

EKSViewPolitique d'Amazon

Cette politique d'accès inclut des autorisations qui permettent à un principal IAM de consulter la plupart des ressources Kubernetes.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
`apps`	`controllerrevisions` , `daemonsets` , `daemonsets/status` , `deployments` , `deployments/scale` , `deployments/status` , `replicasets` , `replicasets/scale` , `replicasets/status` , `statefulsets` , `statefulsets/scale` , `statefulsets/status`	`get`, `list`, `watch`
`autoscaling`	`horizontalpodautoscalers` , `horizontalpodautoscalers/status`	`get`, `list`, `watch`
`batch`	`cronjobs`, `cronjobs/status` , `jobs`, `jobs/status`	`get`, `list`, `watch`
`discovery.k8s.io`	`endpointslices`	`get`, `list`, `watch`
`extensions`	`daemonsets` , `daemonsets/status` , `deployments` , `deployments/scale` , `deployments/status` , `ingresses` , `ingresses/status` , `networkpolicies` , `replicasets` , `replicasets/scale` , `replicasets/status`	`get`, `list`, `watch`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes Kubernetes (autorisations)
	, replicationcontrollers/scale	
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings,events,limitranges , namespaces/statuspods/log,pods/status ,replicationcontrollers/status ,resourcequotas , ouesourcequotas/status	get, list, watch
	namespaces	get, list, watch

Amazon EKSAuto NodePolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy`

Cette politique inclut les autorisations suivantes qui permettent aux composants Amazon EKS d'effectuer les tâches suivantes :

- `kube-proxy`— Surveillez les points de terminaison et les services du réseau, et gérez les événements connexes. Cela permet d'activer la fonctionnalité de proxy réseau à l'échelle du cluster.
- `ipamd`— Gérez les ressources réseau AWS VPC et les interfaces réseau de conteneurs (CNI). Cela permet au démon de gestion des adresses IP de gérer le réseau des pods.
- `coredns`— Accédez à des ressources de découverte de services telles que les terminaux et les services. Cela permet la résolution DNS au sein du cluster.
- `ebs-csi-driver`— Travaillez avec des ressources liées au stockage pour les volumes Amazon EBS. Cela permet le provisionnement et l'attachement dynamiques de volumes persistants.
- `neuron`— Surveillez les nœuds et les pods des appareils AWS Neuron. Cela permet de gérer les accélérateurs AWS Inferentia et Trainium.
- `node-monitoring-agent`— Diagnostic et événements du nœud d'accès. Cela permet de surveiller l'état du cluster et de collecter des diagnostics.

Chaque composant utilise un compte de service dédié et est limité aux seules autorisations requises pour sa fonction spécifique.

Si vous spécifiez manuellement un rôle Node IAM dans un NodeClass, vous devez créer une entrée d'accès qui associe le nouveau rôle Node IAM à cette politique d'accès.

Amazon EKSBlock StoragePolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStoragePolicy`

Cette politique inclut des autorisations qui permettent à Amazon EKS de gérer les ressources d'élection et de coordination des dirigeants pour les opérations de stockage :

- `coordination.k8s.io`— Créez et gérez des objets de location pour l'élection des dirigeants. Cela permet aux composants de stockage EKS de coordonner leurs activités au sein du cluster par le biais d'un mécanisme d'élection du leader.

La politique s'applique aux ressources de location spécifiques utilisées par les composants de stockage EKS afin d'éviter les conflits d'accès aux autres ressources de coordination du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité de stockage par blocs fonctionne correctement.

Amazon EKSLoad BalancingPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy`

Cette politique inclut des autorisations qui permettent à Amazon EKS de gérer les ressources relatives à l'élection des dirigeants à des fins d'équilibrage de charge :

- `coordination.k8s.io`— Créez et gérez des objets de location pour l'élection des dirigeants. Cela permet aux composants d'équilibrage de charge d'EKS de coordonner les activités entre plusieurs répliques en élisant un leader.

La politique vise spécifiquement à équilibrer la charge des ressources de location afin de garantir une bonne coordination tout en empêchant l'accès aux autres ressources de location du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité réseau fonctionne correctement.

EKSNetworkingPolitique d'Amazon

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

Cette politique inclut des autorisations qui permettent à Amazon EKS de gérer les ressources relatives à l'élection des dirigeants à des fins de mise en réseau :

- `coordination.k8s.io`— Créez et gérez des objets de location pour l'élection des dirigeants. Cela permet aux composants réseau EKS de coordonner les activités d'attribution d'adresses IP en élisant un leader.

La politique s'applique spécifiquement à la mise en réseau des ressources de location afin d'assurer une bonne coordination tout en empêchant l'accès aux autres ressources de location du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité réseau fonctionne correctement.

EKSComputePolitique d'Amazon

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

Cette politique inclut des autorisations qui permettent à Amazon EKS de gérer les ressources relatives à l'élection des dirigeants pour les opérations de calcul :

- `coordination.k8s.io`— Créez et gérez des objets de location pour l'élection des dirigeants. Cela permet aux composants informatiques d'EKS de coordonner les activités de dimensionnement des nœuds en élisant un leader.

La politique vise spécifiquement à calculer les ressources de location de gestion tout en autorisant un accès en lecture de base (`get`,`watch`) à toutes les ressources de location du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité réseau fonctionne correctement.

Amazon EKSBlock StorageClusterPolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStorageClusterPolicy`

Cette politique accorde les autorisations nécessaires à la capacité de stockage par blocs d'Amazon EKS Auto Mode. Il permet une gestion efficace des ressources de stockage par blocs au sein des clusters Amazon EKS. La politique inclut les autorisations suivantes :

Gestion des pilotes CSI :

- Créez, lisez, mettez à jour et supprimez des pilotes CSI, en particulier pour le stockage par blocs.

Gestion des volumes :

- Répertoriez, surveillez, créez, mettez à jour, corrigez et supprimez les volumes persistants.
- Répertoriez, surveillez et mettez à jour les réclamations de volume persistantes.
- Corrigez les états des demandes de volume persistantes.

Interaction entre le nœud et le pod :

- Lisez les informations sur le nœud et le pod.
- Gérez les événements liés aux opérations de stockage.

Classes et attributs de stockage :

- Lisez les classes de stockage et les nœuds CSI.
- Lisez les classes d'attributs de volume.

Pièces jointes au volume :

- Répertoriez, observez et modifiez les pièces jointes aux volumes et leur statut.

Opérations de capture instantanée :

- Gérez les instantanés de volume, le contenu des instantanés et les classes de clichés.
- Gérez les opérations relatives aux instantanés de groupes de volumes et aux ressources associées.

Cette politique est conçue pour prendre en charge la gestion complète du stockage par blocs au sein des clusters Amazon EKS exécutés en mode automatique. Il combine des autorisations pour diverses opérations, notamment le provisionnement, l'attachement, le redimensionnement et la capture instantanée de volumes de stockage par blocs.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité de stockage par blocs fonctionne correctement.

Amazon EKSCompute ClusterPolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputeClusterPolicy`

Cette politique accorde les autorisations nécessaires à la fonctionnalité de gestion de calcul d'Amazon EKS Auto Mode. Il permet une orchestration et une mise à l'échelle efficaces des ressources de calcul au sein des clusters Amazon EKS. La politique inclut les autorisations suivantes :

Gestion des nœuds :

- Créez, lisez, mettez à jour, supprimez et gérez le statut de NodePools et NodeClaims.
- Gérez NodeClasses, y compris la création, la modification et la suppression.

Planification et gestion des ressources :

- Accès en lecture aux pods, aux nœuds, aux volumes persistants, aux réclamations de volumes persistants, aux contrôleurs de réplication et aux espaces de noms.
- Accès en lecture aux classes de stockage, aux nœuds CSI et aux pièces jointes aux volumes.
- Répertoriez et surveillez les déploiements, les ensembles de démons, les ensembles de répliques et les ensembles dynamiques.
- Lisez les budgets relatifs aux interruptions de service des pods.

Gestion des événements :

- Créez, lisez et gérez des événements de cluster.

Déprovisionnement des nœuds et expulsion des pods :

- Mettez à jour, corrigez et supprimez des nœuds.
- Créez des expulsions de pods et supprimez des pods si nécessaire.

Gestion des définitions de ressources personnalisées (CRD) :

- Créez un nouveau CRDs.
- Gérez les CRDs éléments spécifiques liés à la gestion des nœuds (NodeClasses NodePools NodeClaims,, et NodeDiagnostics).

Cette politique est conçue pour prendre en charge la gestion complète du calcul au sein des clusters Amazon EKS exécutés en mode automatique. Il combine les autorisations pour diverses opérations, notamment le provisionnement des nœuds, la planification, le dimensionnement et l'optimisation des ressources.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité de gestion du calcul fonctionne correctement.

Amazon EKSLoad BalancingClusterPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

Cette politique accorde les autorisations nécessaires à la fonctionnalité d'équilibrage de charge d'Amazon EKS Auto Mode. Il permet une gestion et une configuration efficaces des ressources d'équilibrage de charge au sein des clusters Amazon EKS. La politique inclut les autorisations suivantes :

Gestion des événements et des ressources :

- Créez et corrigez des événements.
- Accès en lecture aux pods, aux nœuds, aux points de terminaison et aux espaces de noms.

- Mettez à jour le statut des pods.

Gestion des services et des entrées :

- Gestion complète des services et de leurs statuts.
- Contrôle complet des entrées et de leur statut.
- Accès en lecture aux tranches de point de terminaison et aux classes d'entrée.

Liaisons pour les groupes cibles :

- Créez et modifiez les liaisons de groupes cibles et leurs statuts.
- Accès en lecture aux paramètres de la classe d'entrée.

Gestion des définitions de ressources personnalisées (CRD) :

- Créez et lisez tout CRDs.
- Gestion spécifique de targetgroupbindings.eks.amazonaws.com et ingressclassparams.eks.amazonaws.com. CRDs

Configuration du webhook :

- Créez et lisez des configurations de webhook mutant et validant.
- Gérez la eks-load-balancing-webhook configuration.

Cette politique est conçue pour prendre en charge la gestion complète de l'équilibrage de charge au sein des clusters Amazon EKS exécutés en mode automatique. Il combine des autorisations pour diverses opérations, notamment l'exposition aux services, le routage d'entrée et l'intégration aux services d'équilibrage de AWS charge.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité d'équilibrage de charge fonctionne correctement.

Amazon EKSNetworking ClusterPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

Amazon EKSNetworking ClusterPolicy

Cette politique accorde les autorisations nécessaires à la fonctionnalité réseau d'Amazon EKS Auto Mode. Il permet une gestion et une configuration efficaces des ressources réseau au sein des clusters Amazon EKS. La politique inclut les autorisations suivantes :

Gestion des nœuds et des pods :

- Accès en lecture à NodeClasses et à leurs statuts.
- Accès en lecture à NodeClaims et à leurs statuts.
- Accès en lecture aux modules.

Gestion des nœuds CNI :

- Les autorisations CNINodes et leur statut, y compris la création, la lecture, la mise à jour, la suppression et le correctif.

Gestion des définitions de ressources personnalisées (CRD) :

- Créez et lisez tout CRDs.
- Gestion spécifique (mise à jour, correctif, suppression) du CRD cninodes.eks.amazonaws.com.

Gestion des événements :

- Créez et corrigez des événements.

Cette politique est conçue pour prendre en charge la gestion complète du réseau au sein des clusters Amazon EKS exécutés en mode automatique. Il combine les autorisations pour diverses opérations, notamment la configuration du réseau des nœuds, la gestion CNI (Container Network Interface) et la gestion personnalisée des ressources associées.

La politique permet aux composants réseau d'interagir avec les ressources liées aux nœuds, de gérer les configurations de nœuds spécifiques au CNI et de gérer les ressources personnalisées essentielles aux opérations réseau dans le cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires sont en place pour que la capacité réseau fonctionne correctement.

EKSHybridPolitique d'Amazon

> **Note**
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

Cette politique d'accès inclut des autorisations qui accordent à EKS l'accès aux nœuds d'un cluster. Lorsqu'elle est associée à une entrée d'accès, son étendue d'accès est généralement le cluster, plutôt qu'un espace de noms Kubernetes. Cette politique est utilisée par les nœuds hybrides Amazon EKS.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

Groupes d'API Kubernetes	Kubernetes Non-Resource URLs	Ressources Kubernetes	Verbes Kubernetes (autorisations)
*		nodes	list

Amazon EKSCluster InsightsPolicy

> **Note**
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

Cette politique accorde des autorisations en lecture seule pour la fonctionnalité Amazon EKS Cluster Insights. La politique inclut les autorisations suivantes :

Accès aux nœuds : - Répertorier et afficher les nœuds du cluster - Lire les informations sur l'état des nœuds

DaemonSet Accès : - Accès en lecture à la configuration de kube-proxy

Cette politique est automatiquement gérée par le service EKS pour Cluster Insights. Pour de plus amples informations, veuillez consulter the section called "Informations sur le cluster".

Mises à jour des stratégies d'accès

Affichez les détails sur les mises à jour des stratégies d'accès depuis leur introduction. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS enHistorique du document.

Modification	Description	Date
Ajouter une politique pour EKS Cluster Insights	Publier `AmazonEKSClusterInsightsPolicy`	2 décembre 2024
Ajouter des politiques pour Amazon EKS Hybrid	Publier `AmazonEKSHybridPolicy`	2 décembre 2024
Ajouter des politiques pour le mode automatique d'Amazon EKS	Ces politiques d'accès autorisent le rôle IAM du cluster et le rôle IAM du nœud à appeler Kubernetes. APIs AWS les utilise pour automatiser les tâches de routine relatives aux ressources de stockage, de calcul et de réseau.	2 décembre 2024
Addition `AmazonEKSAdminViewPolicy`	Ajoutez une nouvelle politique pour un accès étendu aux vues, y compris à des ressources telles que les secrets.	23 avril 2024
Stratégies d'accès introduites.	Amazon EKS a introduit des stratégies d'accès.	29 mai 2023

Modifier le mode d'authentification pour utiliser les entrées d'accès

Pour commencer à utiliser les entrées d'accès, vous devez remplacer le mode d'authentification du cluster par les modes `API_AND_CONFIG_MAP` ou `API`. Cela ajoute l'API pour les entrées d'accès.

AWS Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour lequel vous souhaitez créer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Le mode d'authentification indique le mode d'authentification actuel du cluster. Si le mode indique API EKS, vous pouvez déjà ajouter des entrées d'accès et vous pouvez ignorer les étapes restantes.
5. Choisissez Gérer l'accès.
6. Pour le mode d'authentification du cluster, sélectionnez un mode avec l'API EKS. Notez que vous ne pouvez pas rétablir le mode d'authentification en un mode qui supprime l'API EKS et les entrées d'accès.
7. Sélectionnez Enregistrer les modifications. Amazon EKS commence à mettre à jour le cluster, le statut du cluster passe à Mise à jour, et la modification est enregistrée dans l'onglet Historique des mises à jour.
8. Attendez que le statut du cluster revienne à Actif. Lorsque le cluster est actif, vous pouvez suivre les étapes décrites the section called "Création d'entrées d'accès" pour ajouter l'accès au cluster pour les principaux IAM.

AWS CLI

1. Installez la AWS CLI, comme décrit dans la section Installation du guide de l'utilisateur de l'interface de ligne de AWS commande.
2. Exécutez la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster. Si vous souhaitez désactiver définitivement la méthode `ConfigMap`, remplacez `API_AND_CONFIG_MAP` par `API`.

   Amazon EKS commence à mettre à jour le cluster, le statut du cluster passe à UPDATING, et le changement est enregistré dans la liste des mises à jour d'aws eks.

```
aws eks update-cluster-config --name my-cluster --access-config
 authenticationMode=API_AND_CONFIG_MAP
```

3. Attendez que le statut du cluster revienne à Actif. Lorsque le cluster est actif, vous pouvez suivre les étapes décrites the section called "Création d'entrées d'accès" pour ajouter l'accès au cluster pour les principaux IAM.

Version de plateforme requise

Pour utiliser les entrées d'accès, le cluster doit disposer d'une version de plate-forme identique ou ultérieure à la version répertoriée dans le tableau suivant, ou d'une version de Kubernetes ultérieure aux versions répertoriées dans le tableau. Si votre version de Kubernetes n'est pas répertoriée, toutes les versions de la plateforme prennent en charge les entrées d'accès.

Version de Kubernetes	Version de la plateforme
1.30	eks.2
1.29	eks.1
1.28	eks.6
1.27	eks.10
1.26	eks.11

Pour plus d'informations, consultez eks/latest/userguide/platform -versions.html [platform-versions, type="documentation "].

Création d'entrées d'accès

Avant de créer des entrées d'accès, tenez compte des éléments suivants :

- Un mode d'authentification correctement défini. Consultez the section called "Mode d'authentification".
- Une entrée d'accès inclut l'Amazon Resource Name (ARN) d'un et d'un seul principal IAM existant. Un principal IAM ne peut pas être inclus dans plus d'une entrée d'accès. Considérations supplémentaires concernant l'ARN que vous spécifiez :

- Les bonnes pratiques IAM recommandent d'accéder à votre cluster en utilisant des rôles IAM dotés d'informations d'identification à court terme, plutôt que des utilisateurs IAM dotés d'informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.
- Si l'ARN est destiné à un rôle IAM, il peut inclure un chemin. ARNs dans `aws-auth` `ConfigMap` les entrées, impossible d'inclure un chemin. Par exemple, votre ARN peut être `arn:aws:iam::<111122223333>:role/<development/apps/my-role>` ou `arn:aws:iam::<111122223333>:role/<my-role>`.
- Si le type de l'entrée d'accès est autre que `STANDARD` (voir la prochaine considération concernant les types), l'ARN doit se trouver dans le même AWS compte que celui de votre cluster. Si c'est le cas`STANDARD`, l'ARN peut se trouver dans le même compte ou dans un AWS compte différent de celui dans lequel se trouve votre cluster.
- Vous ne pouvez pas modifier le principal IAM une fois l'entrée d'accès créée.
- Si vous supprimez le principal IAM avec cet ARN, l'entrée d'accès n'est pas automatiquement supprimée. Nous vous recommandons de supprimer l'entrée d'accès avec un ARN pour un principal IAM que vous supprimez. Si vous ne supprimez pas l'entrée d'accès et que vous recréez le principal IAM, même s'il possède le même ARN, l'entrée d'accès ne fonctionnera pas. En effet, même si l'ARN est le même pour le principal IAM recréé, le `roleID` or `userID` (vous pouvez le constater avec la commande `aws sts get-caller-identity` AWS CLI) est différent pour le principal IAM recréé de ce qu'il était pour le principal IAM d'origine. Même si vous ne voyez pas l'entrée d'accès principale `roleID` ou `userID` d'accès IAM, Amazon EKS la stocke avec l'entrée d'accès.

- Chaque entrée d'accès possède un type. Le type de l'entrée d'accès dépend du type de ressource à laquelle elle est associée et ne définit pas les autorisations. Si vous ne spécifiez aucun type, Amazon EKS définit automatiquement le type sur `STANDARD`
  - `EC2_LINUX`- Pour un rôle IAM utilisé avec des nœuds autogérés Linux ou Bottlerocket
  - `EC2_WINDOWS`- Pour un rôle IAM utilisé avec des nœuds autogérés Windows
  - `FARGATE_LINUX`- Pour un rôle IAM utilisé avec Fargate (AWS Fargate)
  - `HYBRID_LINUX`- Pour un rôle IAM utilisé avec des nœuds hybrides
  - `STANDARD`- Type par défaut si aucun n'est spécifié
  - `EC2`- Pour les classes de nœuds personnalisées en mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Créer une entrée d'accès à une classe de nœud".

  - Vous ne pouvez pas modifier le type une fois l'entrée d'accès créée.
- Il n'est pas nécessaire de créer une entrée d'accès pour un rôle IAM utilisé pour un groupe de nœuds gérés ou un profil Fargate. EKS créera des entrées d'accès (si elles sont activées) ou mettra à jour la carte de configuration d'authentification (si les entrées d'accès ne sont pas disponibles)
- Si le type d'entrée d'accès est `STANDARD`, vous pouvez spécifier un nom d'utilisateur pour l'entrée d'accès. Si vous ne spécifiez aucune valeur pour le nom d'utilisateur, Amazon EKS définit l'une des valeurs suivantes pour vous, en fonction du type d'entrée d'accès et du fait que le principal IAM que vous avez spécifié est un rôle IAM ou un utilisateur IAM. À moins que vous n'ayez une raison spécifique de spécifier votre propre nom d'utilisateur, nous vous recommandons de ne pas en indiquer un et de laisser Amazon EKS le générer automatiquement pour vous. Si vous spécifiez votre propre nom d'utilisateur :
  - Cela ne peut pas commencer par `system:eks:`,`aws:`,`amazon:`, `ouiam:`.
  - Si le nom d'utilisateur correspond à un rôle IAM, nous vous recommandons d'ajouter `{{SessionName}}` ou de le placer `{{SessionNameRaw}}` à la fin de votre nom d'utilisateur. Si vous ajoutez l'un `{{SessionName}}` ou `{{SessionNameRaw}}` l'autre à votre nom d'utilisateur, celui-ci doit inclure deux points avant {{SessionName}}. Lorsque ce rôle est assumé, le nom de session AWS STS spécifié lors de l'attribution du rôle est automatiquement transmis au cluster et apparaît dans CloudTrail les journaux. Par exemple, vous ne pouvez pas avoir le nom d'utilisateur de`john{{SessionName}}`. Le nom d'utilisateur doit être `:john{{SessionName}}` ou `jo:hn{{SessionName}}`. Il suffit que les deux-points soient devant `{{SessionName}}`. Le nom d'utilisateur généré par Amazon EKS dans le tableau suivant inclut un ARN. Comme un ARN inclut les deux-points, il répond à cette exigence. Les deux points ne sont pas obligatoires si vous ne les incluez pas `{{SessionName}}` dans votre nom d'utilisateur. Notez que dans `{{SessionName}}` le caractère spécial, « @ » est remplacé par « - » dans le nom de la session. `{{SessionNameRaw}}`conserve tous les caractères spéciaux dans le nom de session.

Type du principal IAM	Type	Valeur du nom d'utilisateur définie automatiquement par Amazon EKS
Utilisateur	`STANDARD`	ARN de l'utilisateur IAM. Exemple : `arn:aws:`

Type du principal IAM	Type	Valeur du nom d'utilisateur définie automatiquement par Amazon EKS
		`iam::<111122223333>:user/<my-user>`
Rôle	`STANDARD`	L'ARN STS du rôle lorsqu'il est assumé. Amazon EKS ajoute `{{SessionName}}` au rôle. Exemple : `arn:aws:sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}` Si l'ARN du rôle que vous avez spécifié contenait un chemin, Amazon EKS le supprime dans le nom d'utilisateur généré.
Rôle	`EC2_LINUX` ou `EC2_Windows`	`system:node:{{EC2PrivateDNSName}}`
Rôle	`FARGATE_LINUX`	`system:node:{{SessionName}}`
Rôle	`HYBRID_LINUX`	`system:node:{{SessionName}}`

Vous ne pouvez pas modifier le nom d'utilisateur une fois l'entrée d'accès créée.

- Si le type d'entrée d'accès est `STANDARD` et que vous souhaitez utiliser l'autorisation Kubernetes RBAC, vous pouvez ajouter un ou plusieurs noms de groupe à l'entrée d'accès. Après avoir créé une entrée d'accès, vous pouvez ajouter et supprimer des noms de groupes. Pour que le principal IAM ait accès aux objets Kubernetes de votre cluster, vous devez créer et gérer des objets

d'autorisation basée sur les rôles (RBAC) Kubernetes. Créez des Kubernetes `RoleBinding` ou des `ClusterRoleBinding` objets sur votre cluster qui spécifient le nom du groupe sous forme de for. `subject kind: Group` Kubernetes autorise l'accès principal IAM à tous les objets de cluster que vous avez spécifiés dans un Kubernetes `Role` ou à tout `ClusterRole` objet que vous avez également spécifié dans votre liaison. `roleRef` Si vous spécifiez des noms de groupes, nous vous recommandons de vous familiariser avec les objets d'autorisation basée sur les rôles (RBAC) Kubernetes. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

> **Important**
>
> Amazon EKS ne confirme pas que les objets Kubernetes RBAC existant sur votre cluster incluent les noms de groupe que vous spécifiez. Par exemple, si vous créez une entrée d'accès pour un groupe qui n'existe pas actuellement, EKS créera le groupe au lieu de renvoyer une erreur.

Au lieu ou en plus de Kubernetes autorisant l'accès principal IAM aux objets Kubernetes de votre cluster, vous pouvez associer les politiques d'accès Amazon EKS à une entrée d'accès. Amazon EKS autorise les principaux IAM à accéder aux objets Kubernetes de votre cluster avec les autorisations définies dans la politique d'accès. Vous pouvez définir les autorisations d'une politique d'accès aux espaces de noms Kubernetes que vous spécifiez. L'utilisation de politiques d'accès ne vous oblige pas à gérer les objets Kubernetes RBAC. Pour de plus amples informations, veuillez consulter the section called "Politiques d'accès associées".

- Si vous créez une entrée d'accès avec le type `EC2_LINUX` ou `EC2_Windows`, le principal IAM qui crée l'entrée d'accès doit avoir l'autorisation `iam:PassRole`. Pour plus d'informations, consultez la section Octroi à un utilisateur des autorisations lui permettant de transmettre un rôle à un AWS service dans le Guide de l'utilisateur IAM.
- À l'instar du comportement IAM standard, la création et les mises à jour des entrées d'accès sont finalement cohérentes et prennent effet au bout de plusieurs secondes après que l'appel API initial a été renvoyé avec succès. Vous devez concevoir vos applications de sorte qu'elles tiennent compte de ces retards potentiels. Nous vous recommandons de ne pas inclure les créations ou les mises à jour d'entrées d'accès dans les chemins de code critiques à haute disponibilité de votre application. Au lieu de cela, procédez aux modifications dans une routine d'initialisation ou d'installation distincte que vous exécutez moins souvent. Veillez également à vérifier que les modifications ont été propagées avant que les processus de production en dépendent.

- Les entrées d'accès ne prennent pas en charge les rôles liés aux services. Vous ne pouvez pas créer d'entrées d'accès dont l'ARN principal est un rôle lié à un service. Vous pouvez identifier les rôles liés à un service par leur ARN, qui est au format `arn:aws: iam::*:role/aws-service-role/*`.

Vous pouvez créer une entrée d'accès à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour lequel vous souhaitez créer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Choisissez Créer une entrée d'accès.
5. Pour le principal IAM, sélectionnez un utilisateur ou un rôle IAM existant. Les bonnes pratiques IAM recommandent d'accéder à votre cluster en utilisant des rôles IAM dotés d'informations d'identification à court terme, plutôt que des utilisateurs IAM dotés d'informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.
6. Pour Type, si l'entrée d'accès concerne le rôle de nœud utilisé pour les EC2 nœuds Amazon autogérés, sélectionnez EC2 Linux ou EC2 Windows. Dans le cas contraire, acceptez le type par défaut (Standard).
7. Si le type que vous avez choisi est Standard et que vous souhaitez spécifier un nom d'utilisateur, saisissez-le.
8. Si le type que vous avez choisi est Standard et que vous souhaitez utiliser l'autorisation Kubernetes RBAC pour le principal IAM, spécifiez un ou plusieurs noms pour les groupes. Si vous ne spécifiez aucun nom de groupe et que vous souhaitez utiliser l'autorisation Amazon EKS, vous pouvez associer une politique d'accès ultérieurement ou une fois l'entrée d'accès créée.
9. (Facultatif) Pour Balises, attribuez des étiquettes à l'entrée d'accès. Par exemple, pour faciliter la recherche de toutes les ressources portant la même balise.
10. Choisissez Suivant.
11. Sur la page Ajouter une politique d'accès, si le type que vous avez choisi était Standard et que vous souhaitez qu'Amazon EKS autorise le principal IAM à avoir des autorisations sur les objets Kubernetes de votre cluster, procédez comme suit. Sinon, choisissez Next (Suivant).

a. Pour Nom de la politique, choisissez une stratégie d'accès. Vous ne pouvez pas consulter les autorisations des politiques d'accès, mais elles incluent des autorisations similaires à celles des objets Kubernetes destinés aux utilisateurs. `ClusterRole` Pour plus d'informations, consultez la section Rôles destinés aux utilisateurs dans la documentation de Kubernetes.

b. Choisissez l'une des options suivantes :

- Cluster : choisissez cette option si vous souhaitez qu'Amazon EKS autorise le principal IAM à disposer des autorisations définies dans la politique d'accès pour tous les objets Kubernetes de votre cluster.
- Espace de noms Kubernetes : choisissez cette option si vous souhaitez qu'Amazon EKS autorise le principal IAM à disposer des autorisations définies dans la politique d'accès pour tous les objets Kubernetes d'un espace de noms Kubernetes spécifique de votre cluster. Pour Namespace, entrez le nom de l'espace de noms Kubernetes sur votre cluster. Si vous souhaitez ajouter des espaces de noms supplémentaires, choisissez Ajouter un nouvel espace de noms et entrez son nom.

c. Pour ajouter des politiques supplémentaires, Sélectionnez Ajouter une politique. Vous pouvez définir la portée de chaque politique différemment, mais vous ne pouvez ajouter chaque politique qu'une seule fois.

d. Choisissez Suivant.

12. Vérifiez la configuration de votre entrée d'accès. Si quelque chose semble incorrect, choisissez Précédent pour revenir en arrière et corriger l'erreur. Si la configuration est correcte, choisissez Créer.

AWS CLI

1. Installez la AWS CLI, comme décrit dans la section Installation du guide de l'utilisateur de l'interface de ligne de AWS commande.
2. Pour créer une entrée d'accès Vous pouvez utiliser l'un des exemples suivants pour créer des entrées d'accès :
   - Créez une entrée d'accès pour un groupe de nœuds Amazon EC2 Linux autogéré. Remplacez-le *`my-cluster`* par le nom de votre cluster, *`111122223333`* par votre ID de AWS compte et *`EKS-my-cluster-self-managed-ng-1`* par le nom du rôle IAM de votre nœud. Si votre groupe de nœuds est un groupe de nœuds Windows, remplacez-le *`EC2_LINUX`* par`EC2_Windows`.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_LINUX
```

Vous ne pouvez pas utiliser l'`--kubernetes-groups`option lorsque vous spécifiez un type autre que`STANDARD`. Vous ne pouvez pas associer de politique d'accès à cette entrée d'accès, car son type est une valeur autre que`STANDARD`.

- Créez une entrée d'accès qui autorise un rôle IAM qui n'est pas utilisé pour un groupe de nœuds EC2 autogéré par Amazon, avec lequel vous souhaitez que Kubernetes autorise l'accès à votre cluster. *`my-cluster`*Remplacez-le par le nom de votre cluster, *`111122223333`* par votre identifiant de AWS compte et *`my-role`* par le nom de votre rôle IAM. *`Viewers`*Remplacez-le par le nom d'un groupe que vous avez spécifié dans un Kubernetes `RoleBinding` ou un `ClusterRoleBinding` objet de votre cluster.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role --type STANDARD --user Viewers --kubernetes-groups
 Viewers
```

- Créez une entrée d'accès qui permet à un utilisateur IAM de s'authentifier auprès de votre cluster. Cet exemple est fourni car c'est possible, bien que les bonnes pratiques IAM recommandent d'accéder à votre cluster en utilisant des rôles IAM dotés d'informations d'identification à court terme, plutôt que des utilisateurs IAM dotés d'informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:user/my-user --type STANDARD --username my-user
```

Si vous souhaitez que cet utilisateur ait plus d'accès à votre cluster que les autorisations associées aux rôles de découverte de l'API Kubernetes, vous devez associer une politique d'accès à l'entrée d'accès, car l'`--kubernetes-groups`option n'est pas utilisée. Pour plus d'informations, consultez la section the section called “Politiques d'accès associées” et les rôles de découverte d'API dans la documentation de Kubernetes.

Mettre à jour les entrées d'accès

Vous pouvez mettre à jour une entrée d'accès à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour lequel vous souhaitez créer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Choisissez l'entrée d'accès que vous voulez mettre à jour.
5. Choisissez Modifier.
6. Pour Nom d'utilisateur, vous pouvez modifier la valeur existante.
7. Pour Groupes, vous pouvez supprimer les noms de groupes existants ou en ajouter de nouveaux. Si les noms de groupes suivants existent, ne les supprimez pas : system:nodes ou system:bootstrappers. La suppression de ces groupes peut entraîner un dysfonctionnement de votre cluster. Si vous ne spécifiez aucun nom de groupe et que vous souhaitez utiliser l'autorisation Amazon EKS, associez une politique d'accès ultérieurement.
8. Pour Balises, vous pouvez attribuer des étiquettes à l'entrée d'accès. Par exemple, pour faciliter la recherche de toutes les ressources portant la même balise. Vous pouvez également supprimer des balises existantes.
9. Sélectionnez Enregistrer les modifications.
10. Si vous souhaitez associer une stratégie d'accès à l'entrée, consultez the section called “Politiques d'accès associées”.

AWS CLI

1. Installez la AWS CLI, comme décrit dans la section Installation du guide de l'utilisateur de l'interface de ligne de AWS commande.
2. Pour mettre à jour une entrée d'accès, *my-cluster* remplacez-la par le nom de votre cluster, *111122223333* par votre ID de AWS compte et *EKS-my-cluster-my-namespace-Viewers* par le nom d'un rôle IAM.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-groups
 Viewers
```

Vous ne pouvez pas utiliser l'`--kubernetes-groups`option si le type de l'entrée d'accès est une valeur autre que`STANDARD`. Vous ne pouvez pas non plus associer une politique d'accès à une entrée d'un type autre que`STANDARD`.

Supprimer les entrées d'accès

Si vous découvrez que vous avez supprimé une entrée d'accès par erreur, vous pouvez toujours la recréer. Si l'entrée d'accès que vous supprimez est associée à des politiques d'accès, les associations sont automatiquement supprimées. Il n'est pas nécessaire de dissocier les politiques d'accès d'une entrée d'accès avant de supprimer cette entrée d'accès.

Vous pouvez supprimer une entrée d'accès à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster dont vous souhaitez supprimer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Dans la liste des entrées d'accès, choisissez l'entrée d'accès que vous souhaitez supprimer.
5. Sélectionnez Delete (Supprimer).
6. Dans la boîte de dialogue de confirmation, choisissez Delete (Supprimer).

AWS CLI

1. Installez la AWS CLI, comme décrit dans la section Installation du guide de l'utilisateur de l'interface de ligne de AWS commande.
2. Pour supprimer une entrée d'accès, *`my-cluster`* remplacez-la par le nom de votre cluster, *`111122223333`* par votre identifiant de AWS compte et *`my-role`* par le nom du rôle IAM auquel vous ne souhaitez plus avoir accès à votre cluster.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/my-role
```

Accordez aux utilisateurs IAM l'accès à Kubernetes avec un ConfigMap

> **Important**
>
> `aws-auth ConfigMap` est obsolète. Pour connaître la méthode recommandée pour gérer l'accès à Kubernetes APIs, consultez. the section called "Accorder des autorisations"

L'accès à votre cluster à l'aide des principes IAM est activé par l'authentificateur AWS IAM pour Kubernetes, qui s'exécute sur le plan de contrôle Amazon EKS. L'authentificateur obtient ses informations de configuration à partir du `aws-auth ConfigMap`. Pour tous les paramètres `aws-auth ConfigMap`, consultez Format de configuration complet sur GitHub.

Ajout de principaux IAM à votre cluster Amazon EKS

Lorsque vous créez un cluster Amazon EKS, le principal IAM qui crée le cluster reçoit automatiquement des `system:masters` autorisations dans la configuration du contrôle d'accès basé sur les rôles (RBAC) du cluster dans le plan de contrôle Amazon EKS. Ce principal n'apparaît dans aucune configuration visible. Assurez-vous donc de savoir quel principal a créé le cluster à l'origine. Pour autoriser des principaux IAM supplémentaires à interagir avec votre cluster, modifiez-le `aws-auth ConfigMap` dans Kubernetes et créez un Kubernetes `rolebinding` ou `clusterrolebinding` avec le nom d'un Kubernetes que vous spécifiez dans le. `group` `aws-auth ConfigMap`

> **Note**
>
> Pour plus d'informations sur la configuration du contrôle d'accès basé sur les rôles (RBAC) de Kubernetes, consultez la section Utilisation de l'autorisation RBAC dans la documentation de Kubernetes.

1. Déterminer les informations d'identification que `kubectl` utilise pour accéder à votre cluster. Sur votre ordinateur, vous pouvez voir quelles informations d'identification `kubectl` utilise avec la commande suivante. *`~/.kube/config`*Remplacez-le par le chemin de votre `kubeconfig` fichier si vous n'utilisez pas le chemin par défaut.

   ```
   cat ~/.kube/config
   ```

L'exemple qui suit illustre un résultat.

```
[...]
contexts:
- context:
    cluster: my-cluster.region-code.eksctl.io
    user: admin@my-cluster.region-code.eksctl.io
  name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
[...]
```

Dans l'exemple de sortie précédent, les informations d'identification d'un utilisateur nommé *admin* sont configurées pour un cluster nommé *my-cluster*. Si c'est l'utilisateur qui a créé le cluster, alors il a déjà accès à votre cluster. Si ce n'est pas l'utilisateur qui a créé le cluster, vous devez effectuer les étapes restantes pour permettre l'accès au cluster pour les autres principaux IAM. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Vous pouvez voir quels autres principaux ont actuellement accès à votre cluster à l'aide de la commande suivante :

```
kubectl describe -n kube-system configmap/aws-auth
```

L'exemple qui suit illustre un résultat.

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws: iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
```

```
====

Events:  <none>
```

L'exemple précédent est un `aws-auth` `ConfigMap` par défaut. Seul le rôle d'instance du nœud a accès au cluster.

2. Assurez-vous que vous disposez de Kubernetes `rolebindings` et/ou de Kubernetes existants `roles` `clusterroles` et `clusterrolebindings` que vous pouvez y mapper les principaux IAM. Pour plus d'informations sur ces ressources, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

   a. Affichez vos `roles` ou `clusterroles` Kubernetes existants. Les `Roles` sont associés à un `namespace`, mais les `clusterroles` sont associés au cluster.

   ```
   kubectl get roles -A
   ```

   ```
   kubectl get clusterroles
   ```

   b. Affichez les détails de tout `role` ou `clusterrole` renvoyé dans la sortie précédente et assurez-vous qu'il dispose des autorisations (`rules`) dont vous souhaitez que vos principaux IAM disposent dans votre cluster.

   Remplacez *`role-name`* par un nom de `role` renvoyé dans la sortie de la commande précédente. Remplacez *`kube-system`* par l'espace de noms du `role`.

   ```
   kubectl describe role role-name -n kube-system
   ```

   Remplacez *`cluster-role-name`* par un nom de `clusterrole` renvoyé dans la sortie de la commande précédente.

   ```
   kubectl describe clusterrole cluster-role-name
   ```

   c. Affichez vos `rolebindings` ou `clusterrolebindings` Kubernetes existants. Les `Rolebindings` sont associés à un `namespace`, mais les `clusterrolebindings` sont associés au cluster.

   ```
   kubectl get rolebindings -A
   ```

```
kubectl get clusterrolebindings
```

d. Affichez les détails de n'importe quel `rolebinding` ou `clusterrolebinding` et confirmez qu'il possède un `role` ou un `clusterrole` de l'étape précédente répertorié comme un `roleRef` et un nom de groupe répertorié pour `subjects`.

   Remplacez *`role-binding-name`* par un nom de `rolebinding` renvoyé dans la sortie de la commande précédente. Remplacez *`kube-system`* avec le `namespace` du `rolebinding`.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

   L'exemple qui suit illustre un résultat.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

   Remplacez *`cluster-role-binding-name`* par un nom de `clusterrolebinding` renvoyé dans la sortie de la commande précédente.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

   L'exemple qui suit illustre un résultat.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
```

```
  - kind: Group
    name: eks-console-dashboard-full-access-group
    apiGroup: rbac.authorization.k8s.io
  roleRef:
    kind: ClusterRole
    name: eks-console-dashboard-full-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

3. Modifiez le `aws-auth` `ConfigMap`. Vous pouvez utiliser un outil tel que `eksctl` pour mettre à jour le `ConfigMap` ou vous pouvez le mettre à jour manuellement en le modifiant.

> **Important**
>
> Nous vous recommandons d'utiliser `eksctl`, ou un autre outil, pour modifier le `ConfigMap`. Pour plus d'informations sur les autres outils que vous pouvez utiliser, consultez la section Utiliser des outils pour apporter des modifications à l'aws-authConfigMap dans les guides de bonnes pratiques Amazon EKS. Un `aws-auth` `ConfigMap` mis en forme incorrectement peut entraîner la perte de l'accès à votre cluster.

- Consultez les étapes pour modifier le fichier configmap avec eksctl.
- Consultez les étapes pour modifier manuellement le configmap.

Modifier Configmap avec Eksctl

1. Vous avez besoin d'`0.210.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Affichez les mappages actuels dans le `ConfigMap`. Remplacez *`my-cluster`* par le nom de votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

L'exemple qui suit illustre un résultat.

```
ARN
            USERNAME                                GROUPS
 ACCOUNT
```

```
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. Ajoutez un mappage pour un rôle. Remplacez *my-role* par le nom de votre rôle. Remplacez *eks-console-dashboard-full-access-group* par le nom du groupe spécifié dans votre Kubernetes `RoleBinding` ou votre objet. `ClusterRoleBinding` Remplacez *111122223333* par votre ID de compte. Vous pouvez remplacer *admin* par n'importe quel nom que vous choisissez.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws: iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> L'ARN du rôle ne peut pas inclure un chemin tel que`role/my-team/developers/my-role`. Le format de l'ARN doit être `arn:aws: iam::`*111122223333*`:role/`*my-role*. Dans cet exemple, `my-team/developers/` doit être supprimé.

L'exemple qui suit illustre un résultat.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "{arn-aws}iam::111122223333:role/my-role" to
 auth ConfigMap
```

4. Ajoutez un mappage pour un utilisateur. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Remplacez *my-user* par votre nom d'utilisateur. Remplacez *eks-console-dashboard-restricted-access-group* par le nom du groupe spécifié dans votre Kubernetes `RoleBinding` ou votre objet. `ClusterRoleBinding` Remplacez *111122223333* par votre ID de compte. Vous pouvez remplacer *my-user* par n'importe quel nom que vous choisissez.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws: iam::111122223333:user/my-user --username my-user --group eks-
console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

L'exemple qui suit illustre un résultat.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Affichez de nouveau les mappages dans le `ConfigMap`.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

L'exemple qui suit illustre un résultat.

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws: iam::111122223333:role/admin
               my-role                                eks-console-dashboard-full-
access-group
arn:aws: iam::111122223333:user/my-user
            my-user                                eks-console-dashboard-restricted-
access-group
```

Modifier manuellement le fichier Configmap

1. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> Si vous recevez un message d'erreur indiquant `Error from server (NotFound): configmaps "aws-auth" not found` « », utilisez la procédure décrite dans Appliquer l'aws-auth ConfigMap à votre cluster pour appliquer le stock. `ConfigMap`

2. Ajoutez vos principaux IAM à la `ConfigMap`. Un groupe IAM n'est pas un principal IAM, il ne peut donc pas être ajouté au. `ConfigMap`
   - Pour ajouter un rôle IAM (par exemple, pour les utilisateurs fédérés) : ajoutez les détails du rôle à la section `mapRoles` du `ConfigMap`, sous `data`. Ajoutez cette section si elle n'existe pas déjà dans le fichier. Chaque entrée prend en charge les paramètres suivants :
     - rolearn : ARN du rôle IAM à ajouter. Cette valeur ne peut pas inclure de chemin. Par exemple, vous ne pouvez pas spécifier un ARN tel que `arn:aws: iam::111122223333:role/my-team/developers/role-name` . L'ARN doit être `arn:aws: iam::111122223333:role/role-name` .
     - username : nom d'utilisateur dans Kubernetes à mapper au rôle IAM.
     - groups : groupe ou liste des groupes Kubernetes auxquels mapper le rôle. Le groupe peut être un groupe par défaut ou un groupe spécifié dans un `clusterrolebinding` ou `rolebinding`. Pour plus d'informations, consultez la section Rôles par défaut et liaisons de rôles dans la documentation de Kubernetes.
   - Pour ajouter un utilisateur IAM : les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Ajoutez les détails de l'utilisateur dans la section `mapUsers` de la `ConfigMap`, sous `data`. Ajoutez cette section si elle n'existe pas déjà dans le fichier. Chaque entrée prend en charge les paramètres suivants :
     - userarn : ARN de l'utilisateur IAM à ajouter.
     - username : nom d'utilisateur dans Kubernetes à mapper à l'utilisateur IAM.
     - groups : groupe ou liste des groupes Kubernetes auxquels mapper l'utilisateur. Le groupe peut être un groupe par défaut ou un groupe spécifié dans un `clusterrolebinding` ou `rolebinding`. Pour plus d'informations, consultez la section Rôles par défaut et liaisons de rôles dans la documentation de Kubernetes.
3. Par exemple, le bloc YAML ci-dessous contient :
   - Une section `mapRoles` qui mappe l'instance du nœud IAM avec des groupes Kubernetes afin que les nœuds puissent s'inscrire eux-mêmes avec le cluster et le rôle IAM `my-console-viewer-role` mappé à un groupe Kubernetes pouvant afficher toutes les ressources Kubernetes pour tous les clusters. Pour obtenir la liste des autorisations de groupe IAM et Kubernetes requises pour le rôle IAM `my-console-viewer-role`, voir the section called "Autorisations requises".
   - `mapUsers`Section qui fait correspondre l'utilisateur `admin` IAM du AWS compte par défaut au groupe `system:masters` Kubernetes et l'`my-user`utilisateur d'un autre AWS compte mappé à un groupe Kubernetes qui peut afficher les ressources Kubernetes pour un espace de noms

spécifique. Pour obtenir la liste des autorisations de groupe IAM et Kubernetes requises pour l'utilisateur IAM `my-user`, voir the section called "Autorisations requises".

Ajoutez ou supprimez des lignes si nécessaire et remplacez toutes les *`example values`* par vos propres valeurs.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws: iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws: iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws: iam::444455556666:user/my-user
      username: my-user
```

4. Enregistrez le fichier et quittez votre éditeur de texte.

Appliquer la **`ConfigMap aws-auth`** à votre cluster

Le `aws-auth` `ConfigMap` est automatiquement créé et appliqué à votre cluster lorsque vous créez un groupe de nœuds géré ou lorsque vous créez un groupe de nœuds à l'aide de `eksctl`. Il est initialement créé pour permettre aux nœuds de rejoindre votre cluster, mais vous utilisez également cette `ConfigMap` pour ajouter un accès RBAC aux principaux IAM. Si vous avez lancé des nœuds

autogérés et que vous ne les avez pas appliqués `aws-auth ConfigMap` à votre cluster, vous pouvez le faire en suivant la procédure suivante.

1. Vérifiez si vous avez déjà appliqué le `aws-authConfigMap`.

   ```
   kubectl describe configmap -n kube-system aws-auth
   ```

   Si vous recevez un message d'erreur indiquant `Error from server (NotFound): configmaps "aws-auth" not found` « », procédez comme suit pour appliquer le stock`ConfigMap`.

2. Téléchargez, modifiez et appliquez le plan de configuration de l' AWS authentificateur.

   a. Téléchargez la mappe de configuration.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
   aws-auth-cm.yaml
   ```

   b. Dans le fichier `aws-auth-cm.yaml`, remplacez `rolearn` par l'Amazon Resource Name (ARN) du rôle IAM associé par vos nœuds. Pour ce faire, utilisez un éditeur de texte ou remplacez *`my-node-instance-role`* et exécutez la commande suivante :

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
   role|' aws-auth-cm.yaml
   ```

   Ne modifiez aucune autre ligne de ce fichier.

   > **Important**
   >
   > L'ARN du rôle ne peut pas inclure un chemin tel que`role/my-team/developers/my-role`. Le format de l'ARN doit être `arn:aws: iam::`*`111122223333`*`:role/`*`my-role`* . Dans cet exemple, `my-team/developers/` doit être supprimé.

   Vous pouvez inspecter les sorties de la AWS CloudFormation pile pour vos groupes de nœuds et rechercher les valeurs suivantes :

   - InstanceRoleARN — Pour les groupes de nœuds créés avec `eksctl`
   - NodeInstanceRole— Pour les groupes de nœuds créés à l'aide de AWS CloudFormation modèles Amazon EKS vendus dans le AWS Management Console

c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

```
kubectl apply -f aws-auth-cm.yaml
```

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

3. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

Saisissez `Ctrl+C` pour revenir à une invite de shell.

Accorder aux utilisateurs l'accès à Kubernetes avec un fournisseur OIDC externe

Amazon EKS prend en charge l'utilisation des fournisseurs d'identité OpenID Connect (OIDC) comme méthode d'authentification des utilisateurs de votre cluster. Les fournisseurs d'identité OIDC peuvent être utilisés avec ou comme alternative à AWS Identity and Access Management (IAM). Pour plus d'informations sur l'utilisation d'IAM, consultez the section called "Accès à l'API Kubernetes". Après avoir configuré l'authentification dans votre cluster, vous pouvez créer des `roles` et des `clusterroles` Kubernetes pour attribuer des autorisations aux rôles, puis lier les rôles aux identités à l'aide de liaisons `rolebindings` et `clusterrolebindings` Kubernetes. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

- Vous pouvez associer un fournisseur d'identité OIDC à votre cluster.
- Kubernetes ne fournit pas de fournisseur d'identité OIDC. Vous pouvez utiliser un fournisseur d'identité OIDC public existant ou vous exécuter votre propre fournisseur d'identité. Pour obtenir la liste des fournisseurs certifiés, consultez Certification OpenID sur le site OpenID.
- L'URL de l'émetteur du fournisseur d'identité OIDC doit être accessible publiquement, afin qu'Amazon EKS puisse découvrir les clés de signature. Amazon EKS ne prend pas en charge les fournisseurs d'identité OIDC dotés de certificats auto-signés.

- Vous ne pouvez pas désactiver l'authentification IAM sur votre cluster, car elle est toujours requise pour joindre des nœuds à un cluster.
- Un cluster Amazon EKS doit cependant être créé par un principal IAM AWS plutôt que par un utilisateur de fournisseur d'identité OIDC. Cela est dû au fait que le créateur du cluster interagit avec Amazon EKS APIs plutôt qu'avec APIs Kubernetes.
- Les utilisateurs authentifiés par le fournisseur d'identité OIDC sont répertoriés dans le journal d'audit du cluster si les CloudWatch journaux sont activés pour le plan de contrôle. Pour de plus amples informations, veuillez consulter the section called "Activer ou désactiver les journaux du plan de contrôle".
- Vous ne pouvez pas vous connecter au AWS Management Console avec un compte auprès d'un fournisseur OIDC. Vous ne pouvez le faire qu'en vous the section called "Accès aux ressources du cluster" connectant AWS Management Console à l'aide d'un compte AWS Identity and Access Management.

Associer un fournisseur d'identité OIDC

Pour pouvoir associer un fournisseur d'identité OIDC à votre cluster, vous devez obtenir les informations suivantes de votre fournisseur :

URL de l'émetteur

L'URL du fournisseur d'identité OIDC qui permet au serveur d'API de découvrir les clés de signature publiques pour vérifier les jetons. L'URL doit commencer par `https://` et doit correspondre à la `iss` réclamation contenue dans les jetons d'identification OIDC du fournisseur. Conformément à la norme OIDC, les composants du chemin sont autorisés, mais pas les paramètres de requête. Généralement, l'URL se compose uniquement d'un nom d'hôte, comme `https://server.example.org` ou `https://example.com`. Cette URL doit pointer vers le niveau sous `.well-known/openid-configuration` et doit être accessible publiquement sur Internet.

ID client (également connu sous le nom d'audience)

L'ID de l'application cliente qui envoie les demandes d'authentification au fournisseur d'identité OIDC.

Vous pouvez associer un fournisseur d'identité à l'aide de `eksctl` ou de la AWS Management Console.

Associer un fournisseur d'identité à l'aide de eksctl

1. Créez un fichier nommé *associate-identity-provider.yaml* avec les contenus suivants. Remplacez les *example values* par vos propres valeurs. Les valeurs de la section `identityProviders` sont obtenues de votre fournisseur d'identité OIDC. Des valeurs ne sont requises que pour les paramètres `name`, `type`, `issuerUrl` et `clientId` sous `identityProviders`.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

> **Important**
>
> Ne spécifiez pas`system:`, ni aucune partie de cette chaîne, pour `groupsPrefix` ou`usernamePrefix`.

2. Créez le fournisseur.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. À utiliser `kubectl` pour travailler avec votre cluster et votre fournisseur d'identité OIDC, consultez la section Utilisation de kubectl dans la documentation de Kubernetes.

Associer un fournisseur d'identité à l'aide de la AWS console

1. Ouvrez la console Amazon EKS.
2. Sélectionnez votre cluster, puis sélectionnez l'onglet Accès.
3. Dans la section Fournisseurs d'identité OIDC, sélectionnez* Fournisseur d'identité associé*.
4. Sur la page Associate OIDC Identity Provider (Associer un fournisseur d'identité OIDC), saisissez ou sélectionnez les options suivantes, puis sélectionnez Associate (Associer).
   - Dans Nom, saisissez un nom unique pour le fournisseur.
   - Pour URL de l'émetteur, saisissez l'URL de votre fournisseur. Cette URL doit être accessible via Internet.
   - Dans le champ ID client, entrez l'ID client du fournisseur d'identité OIDC (également appelé audience).
   - Pour Nom d'utilisateur, saisissez la revendication à utiliser comme nom d'utilisateur.
   - Pour la réclamation de groupe, entrez la réclamation à utiliser en tant que groupe d'utilisateurs.
   - (Facultatif) Sélectionnez Options avancées, puis saisissez ou sélectionnez les informations suivantes :
     - Nom d'utilisateur : saisissez un préfixe à ajouter aux revendications de nom d'utilisateur. Le préfixe est ajouté aux revendications de nom d'utilisateur pour éviter les conflits avec les noms existants. Si vous ne fournissez aucune valeur et que le nom d'utilisateur est une valeur autre que `email`, le préfixe correspond par défaut à la valeur URL de l'émetteur. Vous pouvez utiliser la valeur `-` pour désactiver toutes les préfixes. Ne spécifiez `system:` aucune partie de cette chaîne.
     - Préfixe de groupes : saisissez un préfixe à ajouter aux revendications de groupes. Le préfixe est ajouté aux revendications de groupe pour éviter les conflits avec des noms existants (tels que `system: groups`). Par exemple, la valeur `oidc:` crée des noms de groupe comme `oidc:engineering` et `oidc:infra`. Ne spécifiez `system:` aucune partie de cette chaîne.
     - Revendications requises : sélectionnez Ajouter une revendication et saisissez une ou plusieurs paires de valeurs clés qui décrivent les revendications requises dans le jeton d'ID de client. Les paires décrivent les demandes requises dans le jeton d'identification. Si elle est définie, la présence de chaque revendication est vérifiée dans le jeton d'ID avec une valeur correspondante.
       a. À utiliser `kubectl` pour travailler avec votre cluster et votre fournisseur d'identité OIDC, consultez la section Utilisation de kubectl dans la documentation de Kubernetes.

Exemple de politique IAM

Si vous voulez empêcher un fournisseur d'identité OIDC d'être associé à un cluster, créez et associez la politique IAM suivante aux comptes IAM de vos administrateurs Amazon EKS. Pour plus d'informations, consultez les sections Création de politiques IAM et ajout d'autorisations d'identité IAM dans le guide de l'utilisateur IAM et Actions dans la référence d'autorisation de service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws: eks:us-west-2.amazonaws.com:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

L'exemple de politique suivant autorise l'association de fournisseur d'identité OIDC si `clientID` est `kubernetes` et `issuerUrl` est `https://cognito-idp.us-west-2amazonaws.com/*` .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
```

```
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

Dissocier un fournisseur d'identité OIDC de votre cluster

Si vous dissociez un fournisseur d'identité OIDC de votre cluster, les utilisateurs inclus dans le fournisseur ne peuvent plus accéder au cluster. Cependant, vous pouvez toujours accéder au cluster avec les principaux IAM.

1. Ouvrez la console Amazon EKS.
2. Dans la section Fournisseurs d'identité OIDC, sélectionnez Dissocier, saisissez le nom du fournisseur d'identité, puis sélectionnez `Disassociate`.

Consultez les ressources Kubernetes dans AWS Management Console

Vous pouvez afficher les ressources Kubernetes déployées sur votre cluster avec la AWS Management Console. Vous ne pouvez pas afficher les ressources Kubernetes avec la AWS CLI

ou eksctl. Pour afficher les ressources Kubernetes à l'aide d'un outil de ligne de commande, utilisez kubectl.

> **Note**
>
> Pour afficher l'onglet Ressources et la section Nœuds de l'onglet Compute du AWS Management Console, le principal IAM que vous utilisez doit disposer d'autorisations IAM et Kubernetes spécifiques. Pour de plus amples informations, veuillez consulter the section called "Autorisations requises".

1. Ouvrez la console Amazon EKS.
2. Dans la liste Clusters, sélectionnez le cluster qui contient les ressources Kubernetes que vous souhaitez afficher.
3. Sélectionnez l'onglet Resources (Ressources).
4. Sélectionnez un groupe Resource type (Type de ressource) pour lequel vous souhaitez afficher les ressources, comme Workloads (Charges de travail). Vous voyez une liste des types de ressources dans ce groupe.
5. Sélectionnez un type de ressource, tel que Deployments (Déploiements), dans le groupe Workloads (Charges de travail). Vous voyez une description du type de ressource, un lien vers la documentation Kubernetes pour plus d'informations sur le type de ressource et une liste des ressources de ce type qui sont déployées sur votre cluster. Si la liste est vide, aucune ressource de ce type n'est déployée sur votre cluster.
6. Sélectionnez une ressource pour afficher plus d'informations à son sujet. Essayez les exemples suivants :
   - Sélectionnez le groupe Workloads (Charges de travail), sélectionnez le type de ressource Deployments (Déploiements), puis sélectionnez la ressource coredns. Lorsque vous sélectionnez une ressource, vous êtes dans Structured view (Vue structurée), par défaut. Pour certains types de ressources, vous voyez une section Pod dans Structured view (Vue structurée). Cette section répertorie les pods gérés par la charge de travail. Vous pouvez sélectionner n'importe quel pod répertorié pour afficher les informations le concernant. Tous les types de ressources n'affichent pas d'informations dans Structured View (Vue structurée). Si vous sélectionnez Raw view (Vue brute) dans le coin supérieur droit de la page de la ressource, vous voyez la réponse JSON complète de l'API Kubernetes pour la ressource.
   - Sélectionnez le groupe Cluster, puis sélectionnez le type de ressource Nodes (Nœuds). Vous voyez une liste de tous les nœuds de votre cluster. Les nœuds peuvent être n'importe

quel type de nœud Amazon EKS. Il s'agit de la même liste que celle que vous voyez dans la section Nodes (Nœuds) lorsque vous sélectionnez l'onglet Compute (Calcul) pour votre cluster. Sélectionnez une ressource de nœud dans la liste. Dans Structured view (Vue structurée), vous voyez également une section Pod. Cette section vous montre tous les pods exécutés sur le nœud.

Autorisations requises

Pour afficher l'onglet Ressources et la section Nœuds de l'onglet Compute du AWS Management Console, le principal IAM que vous utilisez doit disposer d'autorisations IAM et Kubernetes minimales spécifiques. Procédez comme suit pour accorder les autorisations requises à vos principaux IAM.

1. Assurez-vous que les autorisations IAM`eks:AccessKubernetesApi`, ainsi que les autres autorisations IAM nécessaires pour consulter les ressources Kubernetes, sont attribuées au principal IAM que vous utilisez. Pour plus d'informations sur la modification des autorisations d'un principal IAM, consultez la rubrique Contrôle de l'accès des principaux du Guide de l'utilisateur IAM. Pour plus d'informations sur la modification des autorisations pour un rôle, consultez Modification d'une politique d'autorisations de rôle (console) dans le guide de l'utilisateur IAM.

   L'exemple de politique suivant inclut les autorisations nécessaires pour qu'un principal puisse consulter les ressources Kubernetes pour tous les clusters de votre compte. Remplacez *111122223333* par votre identifiant de AWS compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws: ssm:*:111122223333:parameter/*"
        }
    ]
}
```

Pour afficher les nœuds des clusters connectés, le rôle IAM du connecteur Amazon EKS doit être en mesure d'emprunter l'identité du principal dans le cluster. Cela permet au connecteur Amazon EKS de mapper le principal à un utilisateur Kubernetes.

2. Créez une `rolebinding` ou une `clusterrolebinding` Kubernetes qui est liée à un `role` ou un `clusterrole` Kubernetes ou qui dispose des autorisations nécessaires pour afficher les ressources Kubernetes. Pour en savoir plus sur les rôles et les liaisons de rôles Kubernetes, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes. Vous pouvez appliquer l'un des manifestes suivants à votre cluster qui crée un `role` et une `rolebinding` ou un `clusterrole` et une `clusterrolebinding` avec les autorisations Kubernetes nécessaires :

   Afficher les ressources Kubernetes dans tous les espaces de noms

   - Le nom du groupe dans le fichier est `eks-console-dashboard-full-access-group`. Appliquez le manifeste à votre cluster à l'aide de la commande suivante :

     ```
     kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
     ```

   Afficher les ressources Kubernetes dans un espace de noms spécifique

   - L'espace de noms de ce fichier est `default`. Le nom du groupe dans le fichier est `eks-console-dashboard-restricted-access-group`. Appliquez le manifeste à votre cluster à l'aide de la commande suivante :

     ```
     kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
     ```

     Si vous devez modifier le nom du groupe Kubernetes, l'espace de noms, les autorisations ou toute autre configuration dans le fichier, téléchargez le fichier et modifiez-le avant de l'appliquer à votre cluster :

a. Téléchargez le fichier avec l'une des commandes suivantes :

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

b. Modifiez le fichier si nécessaire.

c. Appliquez le manifeste à votre cluster avec l'une des commandes suivantes :

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. Mappez le principal IAM à l'utilisateur ou au groupe Kubernetes dans le. `aws-auth` `ConfigMap` Vous pouvez utiliser un outil tel que `eksctl` pour mettre à jour le `ConfigMap` ou vous pouvez le mettre à jour manuellement en le modifiant.

> **Important**
>
> Nous vous recommandons d'utiliser `eksctl`, ou un autre outil, pour modifier le `ConfigMap`. Pour plus d'informations sur les autres outils que vous pouvez utiliser, consultez la section Utiliser des outils pour apporter des modifications à l'aws-authConfigMap dans les guides de bonnes pratiques Amazon EKS. Un `aws-auth` `ConfigMap` mis en forme incorrectement peut entraîner la perte de l'accès à votre cluster.

Modifier avec eksctl

1. Vous avez besoin d'`0.210.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Affichez les mappages actuels dans le `ConfigMap`. Remplacez *`my-cluster`* par le nom de votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

L'exemple qui suit illustre un résultat.

```
ARN
            USERNAME                                GROUPS
 ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. Ajoutez un mappage pour un rôle. Cet exemple suppose que vous avez attaché les autorisations IAM lors de la première étape à un rôle nommé *my-console-viewer-role*. Remplacez *111122223333* par votre ID de compte.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --group eks-console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> L'ARN du rôle ne peut pas inclure un chemin tel que`role/my-team/developers/my-role`. Le format de l'ARN doit être `arn:aws: iam::`*111122223333*`:role/`*my-role*. Dans cet exemple, `my-team/developers/` doit être supprimé.

L'exemple qui suit illustre un résultat.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws: iam::111122223333:role/my-console-
viewer-role" to auth ConfigMap
```

4. Ajoutez un mappage pour un utilisateur. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Cet exemple suppose que vous avez attaché les autorisations IAM lors de la première étape à un utilisateur nommé *my-user*. Remplacez *111122223333* par votre ID de compte.

```
eksctl create iamidentitymapping \
```

```
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws: iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

L'exemple qui suit illustre un résultat.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Affichez de nouveau les mappages dans le `ConfigMap`.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

L'exemple qui suit illustre un résultat.

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws: iam::111122223333:role/my-console-viewer-role
                                                      eks-console-dashboard-full-
access-group
arn:aws: iam::111122223333:user/my-user
                                                     eks-console-dashboard-restricted-
access-group
```

Modifier ConfigMap manuellement

Pour plus d'informations sur l'ajout d'utilisateurs ou de rôles IAM à la `ConfigMap` `aws-auth`, consultez la rubrique the section called "Ajout de principaux IAM à votre cluster Amazon EKS".

1. Ouvrez le `aws-auth` `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

2. Ajoutez les mappages au `aws-authConfigMap`, mais ne remplacez aucun des mappages existants. L'exemple suivant ajoute des mappages entre les principaux IAM avec des autorisations ajoutées à la première étape et les groupes Kubernetes créés à l'étape précédente :
   - Le rôle *`my-console-viewer-role`* et le `eks-console-dashboard-full-access-group`.
   - L'utilisateur *`my-user`* et le `eks-console-dashboard-restricted-access-group`.

     Ces exemples supposent que vous avez attaché les autorisations IAM au cours de la première étape à un rôle nommé *`my-console-viewer-role`* et à un utilisateur nommé *`my-user`*. Remplacez *`111122223333`* par votre identifiant de AWS compte.

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
    rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws: iam::111122223333:user/my-user
    username: my-user
```

> **Important**
>
> L'ARN du rôle ne peut pas inclure un chemin tel que`role/my-team/developers/my-console-viewer-role`. Le format de l'ARN doit être `arn:aws:iam::`*`111122223333`*`:role/`*`my-console-viewer-role`* . Dans cet exemple, `my-team/developers/` doit être supprimé.

3. Enregistrez le fichier et quittez votre éditeur de texte.

Connect kubectl à un cluster EKS en créant un fichier kubeconfig

Dans cette rubrique, vous allez créer un fichier `kubeconfig` pour votre cluster (ou mettre à jour un fichier existant).

L'outil de ligne de commande `kubectl` utilise les informations de configuration dans les fichiers `kubeconfig` pour communiquer avec le serveur API d'un cluster. Pour plus d'informations, consultez la section Organisation de l'accès au cluster à l'aide de fichiers kubeconfig dans la documentation de Kubernetes.

Amazon EKS utilise la commande `aws eks get-token` avec `kubectl` pour l'authentification du cluster. Par défaut, la AWS CLI utilise les mêmes informations d'identification que celles renvoyées par la commande suivante :

```
aws sts get-caller-identity
```

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called “Configurer kubectl et eksctl”.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un utilisateur IAM ou un rôle IAM avec l'autorisation d'utiliser l'action d'API `eks:DescribeCluster` pour le cluster que vous spécifiez. Pour de plus amples informations, veuillez consulter the section called “Politiques basées sur l'identité”. Si vous utilisez une identité provenant de votre propre fournisseur OpenID Connect pour accéder à votre cluster, consultez la section Utilisation de kubectl dans la documentation de Kubernetes pour créer ou mettre à jour votre fichier. `kube config`

Créer le fichier **kubeconfig** automatiquement

- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Autorisation d'utiliser l'action d'API `eks:DescribeCluster` pour le cluster que vous spécifiez. Pour de plus amples informations, veuillez consulter the section called "Politiques basées sur l'identité".

1. Créez ou mettez à jour un fichier `kubeconfig` pour votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster et remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Par défaut, le fichier de configuration résultant est créé dans le chemin `kubeconfig` par défaut (`.kube`) dans votre répertoire de base ou fusionné avec un fichier `config` existant à cet emplacement. Vous pouvez spécifier un autre chemin avec l'option `--kubeconfig`.

   Vous pouvez spécifier un ARN de rôle IAM avec l'option `--role-arn` à utiliser pour l'authentification lorsque vous émettez des commandes `kubectl`. Dans le cas contraire, le principal IAM de votre chaîne d'identification par défaut de la AWS CLI ou du SDK est utilisé. Vous pouvez consulter l'identité de votre AWS CLI ou de votre SDK par défaut en exécutant la `aws sts get-caller-identity` commande.

   Pour connaître toutes les options disponibles, exécutez la *`aws eks update-kubeconfig help`* commande ou consultez update-kubeconfig dans le manuel de référence des commandes de la CLI AWS .

2. Testez votre configuration.

```
kubectl get svc
```

L'exemple qui suit illustre un résultat.

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

Autoriser les charges de travail Kubernetes à utiliser les comptes de service Kubernetes AWS

A Kubernetes service account provides an identity for processes that run in a Pod. For more information see Gestion des comptes de service in the Kubernetes documentation. If your Pod needs access to AWS services, you can map the service account to an AWS Identity and Access Management identity to grant that access. For more information, see the section called "Identifiants auprès de l'IRSA" or the section called "Identité du pod".

Jetons de compte de service

La BoundServiceAccountTokenVolumefonctionnalité est activée par défaut dans les versions de Kubernetes. Cette fonctionnalité améliore la sécurité des jetons de compte de service en permettant aux charges de travail exécutées sur Kubernetes de demander des jetons Web JSON liés à l'audience, à l'heure et à la clé. Les jetons de compte de service ont une expiration d'une heure. Dans les versions antérieures de Kubernetes, les jetons n'avaient pas d'expiration. Cela signifie que les clients qui exploitent ces jetons doivent actualiser les jetons en moins d'une heure. Les clients Kubernetes suivants SDKs actualisent automatiquement les jetons dans les délais requis :

- Go version `0.15.7` et ultérieure
- Python version `12.0.0` et ultérieure
- Java version `9.0.0` et ultérieure
- JavaScript version `0.10.3` et versions ultérieures
- Branche `master` Ruby

- Haskell version `0.3.0.0`
- Version C# `7.0.5` et versions ultérieures

Si votre charge de travail utilise une version client antérieure, vous devez la mettre à jour. Pour permettre une migration fluide des clients vers les nouveaux jetons de compte de service limités dans le temps, Kubernetes ajoute une période d'expiration prolongée au jeton de compte de service au-delà de la durée par défaut d'une heure. Pour les clusters Amazon EKS, la période d'expiration prolongée est de 90 jours. Le serveur d'API Kubernetes de votre cluster Amazon EKS rejette les demandes contenant des jetons datant de plus de 90 jours. Nous vous recommandons de vérifier vos applications et leurs dépendances pour vous assurer que les versions du client Kubernetes SDKs sont identiques ou ultérieures aux versions répertoriées précédemment.

Lorsque le serveur API reçoit des requêtes avec des jetons de plus d'une heure, il annote l'événement du journal d'audit de l'API avec `annotations.authentication.k8s.io/stale-token`. La valeur de l'annotation ressemble à l'exemple suivant :

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

Si votre cluster a une journalisation de plan de contrôle activée, les annotations se trouvent dans les journaux d'audit. Vous pouvez utiliser la requête CloudWatch Logs Insights suivante pour identifier tous les pods de votre cluster Amazon EKS qui utilisent des jetons périmés :

```
fields @timestamp
|filter @logStream like /kube-apiserver-audit/
|filter @message like /seconds after warning threshold/
|parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

Le `subject` fait référence au compte de service utilisé par le Pod. Le `elapsedtime` indique le temps écoulé (en secondes) après la lecture du dernier jeton. Les demandes adressées au serveur d'API sont refusées lorsque le `elapsedtime` dépasse 90 jours (7 776 000 secondes). Vous devez mettre à jour le SDK client Kubernetes de vos applications de manière proactive afin d'utiliser l'une des versions répertoriées précédemment qui actualise automatiquement le jeton. Si le jeton de compte de service utilisé dure près de 90 jours et que vous n'avez pas suffisamment de temps pour mettre à jour les versions du SDK client avant l'expiration du jeton, vous pouvez résilier les pods existants et en créer de nouveaux. Cela entraîne la récupération du jeton du compte de service, ce qui vous donne 90 jours supplémentaires pour mettre à jour la version de votre client. SDKs

Si le Pod fait partie d'un déploiement, la méthode suggérée pour mettre fin aux Pods tout en conservant une haute disponibilité est d'effectuer un déploiement à l'aide de la commande suivante. Remplacez *my-deployment* par le nom de votre déploiement.

```
kubectl rollout restart deployment/my-deployment
```

Modules complémentaires de cluster

Les modules complémentaires de cluster suivants ont été mis à jour pour utiliser le client Kubernetes SDKs qui récupère automatiquement les jetons des comptes de service. Nous vous recommandons de vous assurer que les versions répertoriées, ou des versions ultérieures, sont installées sur votre cluster.

- Plug-in Amazon VPC CNI pour Kubernetes et versions ultérieures des plugins d'assistance aux métriques. `1.8.0` Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called "CNI Amazon VPC" et cni-metrics-helper.
- Version CoreDNS et versions ultérieures. `1.8.4` Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called "CoreDNS".
- AWS Version du Load Balancer Controller `2.0.0` et versions ultérieures. Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called " AWS Contrôleur Load Balancer".
- Une version actuelle de `kube-proxy`. Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called "kube-proxy".
- AWS pour la version Fluent Bit `2.25.0` ou ultérieure. Pour mettre à jour votre version actuelle, consultez la section Versions publiées le GitHub.
- Version d'image Fluentd 1.14.6-1.2 ou version ultérieure et plug-in de filtre Fluentd pour les métadonnées de Kubernetes version 2.11.1 ou ultérieure.

Octroi AWS d'autorisations Identity and Access Management aux charges de travail sur les clusters Amazon Elastic Kubernetes Service

Amazon EKS propose deux méthodes pour accorder des autorisations AWS Identity and Access Management aux charges de travail exécutées dans des clusters Amazon EKS : les rôles IAM pour les comptes de service et les identités EKS Pod.

Rôles IAM pour les comptes de service

Les rôles IAM pour les comptes de service (IRSA) configurent les applications Kubernetes exécutées AWS avec des autorisations IAM précises pour accéder à diverses autres ressources telles que les compartiments Amazon AWS S3, les tables Amazon DynamoDB, etc. Vous pouvez exécuter plusieurs applications ensemble dans le même cluster Amazon EKS, et vous assurer que chaque application ne dispose que de l'ensemble minimum d'autorisations dont elle a besoin. L'IRSA a été conçu pour prendre en charge diverses options de déploiement de Kubernetes prises AWS en charge par Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on et les clusters Kubernetes autogérés sur AWS les instances Amazon. EC2 Ainsi, l'IRSA a été créée à l'aide d'un AWS service de base tel que IAM, et ne dépendait pas directement du service Amazon EKS et de l'API EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA".

Identités d'EKS Pod

EKS Pod Identity offre aux administrateurs de clusters un flux de travail simplifié pour authentifier les applications afin d'accéder à diverses autres AWS ressources telles que les compartiments Amazon S3, les tables Amazon DynamoDB, etc. L'identité du pod EKS est réservée uniquement à EKS et, par conséquent, elle simplifie la manière dont les administrateurs de cluster peuvent configurer les applications Kubernetes pour obtenir des autorisations IAM. Ces autorisations peuvent désormais être facilement configurées en moins d'étapes directement via AWS Management Console l'API EKS et la AWS CLI, et il n'y a aucune action à effectuer au sein du cluster dans aucun objet Kubernetes. Les administrateurs de clusters n'ont pas besoin de basculer entre les services EKS et IAM, ni d'utiliser des opérations IAM privilégiées pour configurer les autorisations requises par vos applications. Les rôles IAM peuvent désormais être utilisés dans plusieurs clusters sans qu'il soit nécessaire de mettre à jour la politique d'approbation des rôles lors de la création de nouveaux clusters. Les informations d'identification IAM fournies par l'identité du pod EKS comprennent des balises de session de rôle, avec des attributs tels que le nom du cluster, l'espace de noms, le nom du compte de service. Les balises de session de rôle permettent aux administrateurs de créer un rôle unique qui peut fonctionner avec tous les comptes de service en autorisant l'accès aux AWS ressources en fonction des balises correspondantes. Pour de plus amples informations, veuillez consulter the section called "Identité du pod".

Comparaison de l'identité du pod EKS et de l'IRSA

À un niveau élevé, l'identité du pod EKS et l'IRSA vous permettent tous deux d'accorder des autorisations IAM aux applications exécutées sur des clusters Kubernetes. Mais ils sont fondamentalement différents dans la façon dont vous les configurez, les limites prises en charge et les fonctionnalités activées. Nous comparons ci-dessous certains des principaux aspects des deux solutions.

> **Note**
>
> AWS recommande d'utiliser EKS Pod Identities pour accorder l'accès aux AWS ressources à vos pods dans la mesure du possible. Pour de plus amples informations, veuillez consulter the section called "Identité du pod".

Attribut	Identité du pod EKS	IRSA
Extensibilité des rôles	Vous devez configurer chaque rôle une fois pour établir la confiance avec le nouveau principal de service Amazon EKS `pods.eks.amazonaws.com` . Après cette étape unique, il n'est pas nécessaire de mettre à jour la politique de confiance du rôle chaque fois qu'il est utilisé dans un nouveau cluster.	Vous devez mettre à jour la politique de confiance du rôle IAM avec le nouveau point de terminaison du fournisseur OIDC du cluster EKS chaque fois que vous souhaitez utiliser le rôle dans un nouveau cluster.
Capacité de mise à l'échelle des clusters	EKS Pod Identity n'oblige pas les utilisateurs à configurer le fournisseur IAM OIDC. Cette limite ne s'applique donc pas.	Chaque cluster EKS est associé à une URL d'émetteur OpenID Connect (OIDC). Pour utiliser IRSA, un fournisseur OpenID Connect unique doit être créé pour chaque cluster EKS dans IAM. IAM a une limite globale par défaut

Attribut	Identité du pod EKS	IRSA
		de 100 fournisseurs OIDC pour chaque AWS compte. Si vous prévoyez d'avoir plus de 100 clusters EKS pour chaque AWS compte IRSA, vous atteindrez la limite de fournisseurs IAM OIDC.
Capacité de mise à l'échelle des rôles	EKS Pod Identity n'oblige pas les utilisateurs à définir une relation de confiance entre le rôle IAM et le compte de service dans la politique de confiance. Cette limite ne s'applique donc pas.	Dans IRSA, vous définissez la relation de confiance entre un rôle IAM et un compte de service dans la politique de confiance du rôle. Par défaut, la longueur de la politique d'approbation est `2048`. Cela signifie que vous pouvez généralement définir 4 relations d'approbation dans une seule politique d'approbation. Bien qu'il soit possible d'augmenter la limite de longueur de la politique d'approbation, vous êtes généralement limité à un maximum de 8 relations d'approbation au sein d'une même politique d'approbation.

Attribut	Identité du pod EKS	IRSA
Utilisation des quotas de l'API STS	EKS Pod Identity simplifie la transmission des AWS informations d'identification à vos pods et ne nécessite pas que votre code passe des appels directement avec le AWS Security Token Service (STS). Le service EKS gère l'attribution des rôles et fournit des informations d'identif ication aux applications écrites à l'aide du AWS SDK dans vos pods sans que vos pods communiquent avec AWS STS ou n'utilisent le quota d'API STS.	Dans IRSA, les applicati ons écrites à l'aide du AWS SDK dans vos pods utilisent des jetons pour appeler l'`AssumeRoleWithWebIdentity` API sur le AWS Security Token Service (STS). Selon la logique de votre code sur le AWS SDK, il est possible que votre code fasse des appels inutiles à AWS STS et reçoive des erreurs de régulation.

Attribut	Identité du pod EKS	IRSA
Réutilisation des rôles	AWS Les informations d'identification temporaires STS fournies par EKS Pod Identity incluent des balises de session de rôle, telles que le nom du cluster, l'espace de noms, le nom du compte de service. Les balises de session de rôle permettent aux administrateurs de créer un rôle IAM unique qui peut être utilisé avec plusieurs comptes de service, avec des autorisations effectives différentes, en autorisant l'accès aux AWS ressources en fonction des balises qui leur sont associées . C'est ce qu'on appelle le contrôle d'accès par attributs (ABAC). Pour de plus amples informations, veuillez consulter the section called "Autoriser l'accès aux Pods".	AWS Les balises de session STS ne sont pas prises en charge. Vous pouvez réutiliser un rôle entre les clusters, mais chaque pod reçoit toutes les autorisations du rôle.
Environnements pris en charge	L'identité du pod EKS est uniquement disponible sur Amazon EKS.	L'IRSA peut être utilisé comme Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on AWS et les clusters Kubernetes autogérés sur les instances Amazon. EC2

Attribut	Identité du pod EKS	IRSA
Versions EKS prises en charge	Toutes les versions de clusters EKS prises en charge. Pour les versions de plateform e spécifiques, consultez the section called "Versions du cluster de l'identité du pod EKS".	Toutes les versions de clusters EKS prises en charge.

Rôles IAM pour les comptes de service

Les applications situées dans les conteneurs d'un pod peuvent utiliser un AWS SDK ou la AWS CLI pour envoyer des demandes d'API aux AWS services à l'aide des AWS autorisations Identity and Access Management (IAM). Les applications doivent signer leurs demandes AWS d'API avec des AWS informations d'identification. Les rôles IAM pour les comptes de service (IRSA) permettent de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance Amazon fournissent des informations d'identification aux instances Amazon EC2 . Au lieu de créer et de distribuer vos AWS informations d'identification aux conteneurs ou d'utiliser le rôle de l' EC2 instance Amazon, vous associez un rôle IAM à un compte de service Kubernetes et configurez vos Pods pour utiliser le compte de service. Vous ne pouvez pas utiliser de rôles IAM pour des comptes de service dotés de clusters locaux pour Amazon EKS on AWS Outposts.

Les rôles IAM pour les comptes de service offrent les bénéfices suivants :

- Privilège minimal : vous pouvez attribuer des autorisations IAM à un compte de service, et seuls les pods qui utilisent ce compte de service ont accès à ces autorisations. Cette fonctionnalité élimine également le besoin de solutions tierces telles que `kiam` ou `kube2iam`.
- Isolation des informations d'identification : lorsque l'accès à l'Amazon EC2 Instance Metadata Service (IMDS) est restreint, les conteneurs d'un pod peuvent uniquement récupérer les informations d'identification pour le rôle IAM associé au compte de service utilisé par le conteneur. Un conteneur n'a jamais accès aux informations d'identification utilisées par d'autres conteneurs dans d'autres pods. Si l'IMDS n'est pas restreint, les conteneurs du pod ont également accès au rôle IAM du nœud Amazon EKS et les conteneurs peuvent être en mesure d'accéder aux informations d'identification des rôles IAM des autres pods sur le même nœud. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

> **Note**
>
> Les pods configurés avec `hostNetwork: true` auront toujours un accès IMDS, mais la CLI AWS SDKs and utilisera les informations d'identification IRSA lorsqu'elle est activée.

- Auditabilité — La journalisation des accès et des événements est disponible AWS CloudTrail pour garantir un audit rétrospectif.

> **Important**
>
> Les conteneurs ne constituent pas une limite de sécurité, et l'utilisation de rôles IAM pour les comptes de service ne change rien à cela. Les pods affectés au même nœud partageront un noyau et potentiellement d'autres ressources en fonction de la configuration de votre pod. Alors que les pods exécutés sur des nœuds distincts seront isolés au niveau de la couche de calcul, certaines applications de nœuds disposent d'autorisations supplémentaires dans l'API Kubernetes au-delà du cadre d'une instance individuelle. Par exemple `kubeletkube-proxy`, les pilotes de stockage CSI ou vos propres applications Kubernetes.

Activez les rôles IAM pour les comptes de service en suivant les procédures suivantes :

1. Créez un fournisseur IAM OIDC pour votre cluster : vous n'effectuez cette procédure qu'une seule fois pour chaque cluster.

   > **Note**
   >
   > Si vous avez activé le point de terminaison du VPC EKS, le point de terminaison du service EKS OIDC n'est pas accessible depuis ce VPC. Par conséquent, les opérations telles que la création d'un fournisseur OIDC avec `eksctl` dans le VPC n'aboutiront pas et entraîneront une expiration lors de la tentative de demande de `https://oidc.eks.region.amazonaws.com`. Voici un exemple de message d'erreur :
   >
   > ```
   > server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
   > ```
   >
   > Pour terminer cette étape, vous pouvez exécuter la commande en dehors du VPC, par exemple dans AWS CloudShell ou sur un ordinateur connecté à Internet. Vous pouvez

également créer un résolveur conditionnel à horizon partagé dans le VPC, tel que Route 53 Resolver, pour utiliser un résolveur différent pour l'URL de l'émetteur OIDC et ne pas utiliser le DNS du VPC pour celle-ci. Pour un exemple de transfert conditionnel dans CoreDNS, consultez la demande de fonctionnalité Amazon EKS sur. GitHub

2. Attribuer des rôles IAM aux comptes de service Kubernetes : effectuez cette procédure pour chaque ensemble unique d'autorisations que vous souhaitez attribuer à une application.
3. Configurer les pods pour utiliser un compte de service Kubernetes : effectuez cette procédure pour chaque pod devant accéder aux services. AWS
4. Utiliser IRSA avec le AWS SDK : vérifiez que la charge de travail utilise un AWS SDK d'une version prise en charge et qu'elle utilise la chaîne d'identification par défaut.

Informations générales sur IAM, Kubernetes et OpenID Connect (OIDC)

En 2014, AWS Identity and Access Management a ajouté la prise en charge des identités fédérées à l'aide d'OpenID Connect (OIDC). Cette fonctionnalité vous permet d'authentifier les appels d' AWS API auprès des fournisseurs d'identité pris en charge et de recevoir un jeton Web OIDC JSON (JWT) valide. Vous pouvez transmettre ce jeton à l'opération de l'`AssumeRoleWithWebIdentity`API AWS STS et recevoir des informations d'identification de rôle temporaires IAM. Vous pouvez utiliser ces informations d'identification pour interagir avec n'importe quel AWS service, y compris Amazon S3 et DynamoDB.

Chaque jeton JWT est signé par une paire de clés de signature. Les clés sont servies sur le fournisseur OIDC géré par Amazon EKS et la clé privée change tous les 7 jours. Amazon EKS conserve les clés publiques jusqu'à leur expiration. Si vous connectez des clients OIDC externes, sachez que vous devez actualiser les clés de signature avant l'expiration de la clé publique. Apprenez à récupérer les clés de signature pour valider les jetons OIDC.

Kubernetes utilise depuis longtemps les comptes de service comme son propre système d'identité interne. Les pods peuvent s'authentifier auprès du serveur d'API Kubernetes à l'aide d'un jeton monté automatiquement (qui était un JWT non OIDC) que seul le serveur d'API Kubernetes a pu valider. Ces anciens jetons de compte de service n'expirent pas, et la rotation de la clé de signature est un processus difficile. Dans la version Kubernetes`1.12`, le support d'une nouvelle fonctionnalité a été ajouté. `ProjectedServiceAccountToken` Cette fonctionnalité est un jeton Web OIDC JSON qui contient également l'identité du compte de service et prend en charge une audience configurable.

Amazon EKS héberge un point de terminaison de découverte OIDC public pour chaque cluster qui contient les clés de signature des jetons Web `ProjectedServiceAccountToken` JSON afin que les systèmes externes, tels que IAM, puissent valider et accepter les jetons OIDC émis par Kubernetes.

Créer un fournisseur d'identité OIDC IAM pour votre cluster

Une URL d'émetteur OpenID Connect (OIDC) est associée à votre cluster. Pour utiliser les rôles AWS Identity and Access Management (IAM) pour les comptes de service, un fournisseur IAM OIDC doit exister pour l'URL de l'émetteur OIDC de votre cluster.

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called “Configurer kubectl et eksctl”.
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called “Cluster d'accès avec kubectl”.

Vous pouvez créer un fournisseur IAM OIDC pour votre cluster à l'aide `eksctl` du. AWS Management Console

Créer un fournisseur OIDC (eksctl)

1. Version `0.210.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Déterminez l'identifiant de l'émetteur OIDC pour votre cluster.

   Récupérez l'ID d'émetteur OIDC de votre cluster et stockez-le dans une variable. Remplacez `<my-cluster>` par votre propre valeur.

   ```
   cluster_name=<my-cluster>
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   echo $oidc_id
   ```

3. Déterminez si un fournisseur IAM OIDC avec l'ID d'émetteur de votre cluster figure déjà dans votre compte.

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

   Si une sortie est renvoyée, cela signifie que vous disposez déjà d'un fournisseur IAM OIDC pour votre cluster et que vous pouvez ignorer l'étape suivante. Si aucune sortie n'est renvoyée, vous devez créer un fournisseur OIDC IAM pour votre cluster.
4. Créez votre fournisseur d'identité OIDC IAM pour votre cluster avec la commande suivante.

   ```
   eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
   ```

   > **Note**
   >
   > Si vous avez activé le point de terminaison du VPC EKS, le point de terminaison du service EKS OIDC n'est pas accessible depuis ce VPC. Par conséquent, vos opérations telles que la création d'un fournisseur OIDC `eksctl` dans le VPC ne fonctionneront pas et entraîneront un délai d'attente. Voici un exemple de message d'erreur :

   ```
   ** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
   ```

Pour terminer cette étape, vous pouvez exécuter la commande en dehors du VPC, par exemple dans AWS CloudShell ou sur un ordinateur connecté à Internet. Vous pouvez également créer un résolveur conditionnel à horizon partagé dans le VPC, tel que Route 53 Resolver, afin d'utiliser un résolveur différent pour l'URL de l'émetteur OIDC et de ne pas utiliser le DNS du VPC pour celle-ci. Pour un exemple de transfert conditionnel dans CoreDNS, consultez la demande de fonctionnalité Amazon EKS sur. GitHub

Créer un fournisseur OIDC (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de gauche, sélectionnez Clusters, puis sélectionnez le nom de votre cluster sur la page Clusters.
3. Dans la section Détails de l'onglet Présentation, notez la valeur de l'URL du fournisseur d'OpenID Connect.
4. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
5. Dans le panneau de navigation, sélectionnez Identity Providers (Fournisseurs d'identité) sous Access management (Gestion des accès). Si un fournisseur est répertorié et correspond à l'URL de votre cluster, alors vous avez déjà un fournisseur pour votre cluster. Si aucun fournisseur correspondant à l'URL de votre cluster n'est répertorié, vous devez en créer un.
6. Pour créer un fournisseur, cliquez sur Ajouter un fournisseur.
7. Pour le type de fournisseur, sélectionnez OpenID Connect.
8. Dans le champ URL du fournisseur, entrez l'URL du fournisseur OIDC de votre cluster.
9. Dans le champ Audience, entrez `sts.amazonaws.com`.
10. (Facultatif) Ajoutez des balises, par exemple une balise pour identifier le cluster destiné à ce fournisseur.
11. Choisissez Ajouter un fournisseur.

Étape suivante : the section called "Attribuer un rôle IAM"

Attribuer des rôles IAM aux comptes de service Kubernetes

Cette rubrique explique comment configurer un compte de service Kubernetes pour qu'il assume un rôle AWS Identity and Access Management (IAM). Tous les pods configurés pour utiliser le compte de service peuvent ensuite accéder à tout AWS service auquel le rôle est autorisé à accéder.

Prérequis

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un en suivant l'un des guides présentés dansMise en route.
- Un fournisseur IAM OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un ou comment en créer un, consultez the section called "Fournisseur IAM OIDC".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Cluster d'accès avec kubectl".

Étape 1 : créer une politique IAM

Si vous souhaitez associer une politique IAM existante à votre rôle IAM, passez à l'étape suivante.

1. Créez une politique IAM. Vous pouvez créer votre propre politique ou copier une politique AWS gérée qui accorde déjà certaines des autorisations dont vous avez besoin et la personnaliser en fonction de vos besoins spécifiques. Pour plus d'informations, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.
2. Créez un fichier qui inclut les autorisations pour les AWS services auxquels vous souhaitez que vos pods accèdent. Pour obtenir la liste de toutes les actions pour tous les AWS services, consultez la référence d'autorisation des services.

Vous pouvez exécuter la commande suivante pour créer un exemple de fichier de politique autorisant l'accès en lecture seule à un compartiment Amazon S3. Vous pouvez éventuellement stocker des informations de configuration ou un script de démarrage dans ce compartiment, et les conteneurs de votre Pod peuvent lire le fichier depuis le compartiment et le charger dans votre application. Si vous souhaitez créer cet exemple de politique, copiez le contenu suivant sur votre appareil. *my-pod-secrets-bucket*Remplacez-le par le nom de votre compartiment et exécutez la commande.

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

3. Créez la politique IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

Étape 2 : créer et associer un rôle IAM

Créez un rôle IAM et associez-le à un compte de service Kubernetes. Vous pouvez utiliser l'une ou l'autre `eksctl` option ou la AWS CLI.

Créer et associer un rôle (eksctl)

Cette `eksctl` commande crée un compte de service Kubernetes dans l'espace de noms spécifié, crée un rôle IAM (s'il n'existe pas) avec le nom spécifié, attache un ARN de politique IAM existant au rôle et annote le compte de service avec l'ARN du rôle IAM. Assurez-vous de remplacer les exemples de valeurs d'espace réservé dans cette commande par vos valeurs spécifiques. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster
 my-cluster --role-name my-role \
    --attach-policy-arn arn:aws: iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> Si le rôle ou le compte de service existe déjà, la commande précédente peut échouer. `eksctl` propose différentes options que vous pouvez fournir dans ces situations. Pour de plus amples informations, exécutez `eksctl create iamserviceaccount --help`.

Créer et associer un rôle (AWS CLI)

Si vous possédez déjà un compte de service Kubernetes et que vous souhaitez assumer un rôle IAM, vous pouvez ignorer cette étape.

1. Créez un compte de service Kubernetes. Copiez les contenus suivants sur votre appareil. *`my-service-account`*Remplacez-le par le nom de votre choix et *`default`* par un autre espace de noms, si nécessaire. Si vous modifiez*`default`*, l'espace de noms doit déjà exister.

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

2. Définissez votre identifiant de AWS compte sur une variable d'environnement à l'aide de la commande suivante.

   ```
   account_id=$(aws sts get-caller-identity --query "Account" --output text)
   ```

3. Définissez le fournisseur d'identité OIDC de votre cluster sur une variable d'environnement à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
   query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
   ```

4. Définissez des variables pour l'espace de noms et le nom du compte de service. *my-service-account*Remplacez-le par le compte de service Kubernetes dont vous souhaitez assumer le rôle. Remplacez *default* par l'espace de noms du compte de service.

```
export namespace=default
export service_account=my-service-account
```

5. Exécutez la commande suivante pour créer un fichier de stratégie d'approbation pour le rôle IAM. Si vous souhaitez autoriser tous les comptes de service d'un espace de noms à utiliser le rôle, copiez le contenu suivant sur votre appareil. Remplacez *StringEquals* par `StringLike` et remplacez *$service_account* par`*`. Vous pouvez ajouter plusieurs entrées dans les conditions `StringEquals` ou `StringLike` pour autoriser plusieurs comptes de service ou espaces de noms à endosser le rôle. Pour autoriser les rôles provenant d'un AWS compte différent de celui dans lequel se trouve votre cluster à assumer le rôle, consultez the section called "IAM entre comptes" pour plus d'informations.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

6. Créez le rôle. Remplacez *my-role* avec un nom pour votre rôle IAM, et *my-role-description* avec une description de votre rôle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

7. Liez une politique IAM à votre rôle. Remplacez *my-role* par le nom de votre rôle IAM et *my-policy* par le nom d'une politique existante que vous avez créée.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws: iam::
$account_id:policy/my-policy
```

8. Annotez votre compte de service avec l'Amazon Resource Name (ARN) du rôle IAM que vous souhaitez que le compte de service endosse. Remplacez *my-role* par le nom de votre rôle IAM existant. Supposons que vous ayez autorisé un rôle provenant d'un AWS compte différent de celui dans lequel se trouve votre cluster à assumer ce rôle lors d'une étape précédente. Assurez-vous ensuite de spécifier le AWS compte et le rôle de l'autre compte. Pour de plus amples informations, veuillez consulter the section called "IAM entre comptes".

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws: iam::$account_id:role/my-role
```

9. (Facultatif) Configurez le point de terminaison du service AWS Security Token pour un compte de service. AWS recommande d'utiliser un point de terminaison AWS STS régional au lieu du point de terminaison global. Cela réduit la latence, fournit une redondance intégrée et augmente la validité des jetons de session.

Étape 3 : Confirmer la configuration

1. Vérifiez que la politique de confiance du rôle IAM est correctement configurée.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

L'exemple qui suit illustre un résultat.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-service-
account",
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

2. Vérifiez que la stratégie que vous avez associée à votre rôle lors d'une étape précédente est associée au rôle.

```
aws iam list-attached-role-policies --role-name my-role --query
 "AttachedPolicies[].PolicyArn" --output text
```

L'exemple qui suit illustre un résultat.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Définissez une variable pour stocker l'Amazon Resource Name (ARN) de la stratégie que vous souhaitez utiliser. *my-policy*Remplacez-le par le nom de la politique pour laquelle vous souhaitez confirmer les autorisations.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Affichez la version par défaut de la stratégie.

```
aws iam get-policy --policy-arn $policy_arn
```

L'exemple qui suit illustre un résultat.

```
{
    "Policy": {
```

```
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Consultez le contenu de la politique pour vous assurer qu'elle inclut toutes les autorisations dont votre Pod a besoin. Si nécessaire, remplacez *1* la commande suivante par la version renvoyée dans la sortie précédente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

L'exemple qui suit illustre un résultat.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Si vous avez créé l'exemple de stratégie lors d'une étape précédente, le résultat est le même. Si vous avez créé une politique différente, le *example* contenu est différent.

6. Vérifiez que le compte de service Kubernetes est annoté avec le rôle.

```
kubectl describe serviceaccount my-service-account -n default
```

L'exemple qui suit illustre un résultat.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-
role
```

```
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

Étapes suivantes

- the section called "Assigner à Pod"

Configurer les pods pour utiliser un compte de service Kubernetes

Si un Pod doit accéder à AWS des services, vous devez le configurer pour utiliser un compte de service Kubernetes. Le compte de service doit être associé à un rôle AWS Identity and Access Management (IAM) autorisé à accéder aux services. AWS

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un à l'aide de l'un des guides inclus dansMise en route.
- Un fournisseur IAM OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un ou comment en créer un, consultez the section called "Fournisseur IAM OIDC".
- Compte de service Kubernetes existant associé à un rôle IAM. Le compte de service doit être annoté avec l'Amazon Resource Name (ARN) du rôle IAM. Le rôle doit être associé à une politique IAM contenant les autorisations dont vous souhaitez que vos pods disposent pour utiliser les AWS services. Pour plus d'informations sur la façon de créer et de configurer le compte de service et le rôle, consultez the section called "Attribuer un rôle IAM".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la

version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl 1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Cluster d'accès avec kubectl".

1. Utilisez la commande suivante pour créer un manifeste de déploiement avec lequel vous pouvez déployer un Pod pour confirmer la configuration. Remplacez les *`example values`* par vos propres valeurs.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Appliquez le manifeste à votre cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Vérifiez que les variables d'environnement requises existent pour votre Pod.

   a. Affichez les pods qui ont été déployés lors du déploiement à l'étape précédente.

```
kubectl get pods | grep my-app
```

   L'exemple qui suit illustre un résultat.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. Affichez l'ARN du rôle IAM utilisé par le Pod.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
```

L'exemple qui suit illustre un résultat.

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/my-role
```

L'ARN du rôle doit correspondre à l'ARN du rôle avec lequel vous avez annoté le compte de service existant. Pour en savoir plus sur l'annotation du compte de service, consultez the section called "Attribuer un rôle IAM".

c. Vérifiez que le Pod est équipé d'un fichier contenant un jeton d'identité Web.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

L'exemple qui suit illustre un résultat.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/serviceaccount/
token
```

Il `kubelet` demande et stocke le jeton au nom du Pod. Par défaut, `kubelet` actualise le jeton si ce dernier est supérieur à 80 % de son temps total avant le live, ou si le jeton est âgé de plus de 24 heures. Vous pouvez modifier la durée d'expiration de n'importe quel compte autre que le compte de service par défaut en utilisant les paramètres de la spécification de votre Pod. Pour plus d'informations, consultez la section Projection du volume des jetons de compte de service dans la documentation Kubernetes.

Le webhook Amazon EKS Pod Identity sur le cluster surveille les pods qui utilisent un compte de service avec l'annotation suivante :

```
eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-role
```

Le webhook applique les variables d'environnement précédentes à ces pods. Votre cluster n'a pas besoin d'utiliser le webhook pour configurer les variables d'environnement et les

montages de fichiers de jetons. Vous pouvez configurer manuellement les pods pour qu'ils contiennent ces variables d'environnement. Les versions prises en charge du AWS SDK recherchent d'abord ces variables d'environnement dans le fournisseur de chaîne d'informations d'identification. Les informations d'identification du rôle sont utilisées pour les pods qui répondent à ces critères.

4. Vérifiez que vos Pods peuvent interagir avec les AWS services à l'aide des autorisations que vous avez attribuées dans la politique IAM associée à votre rôle.

> **Note**
>
> Lorsqu'un pod utilise les AWS informations d'identification d'un rôle IAM associé à un compte de service, la AWS CLI ou un autre SDKs élément des conteneurs de ce pod utilisent les informations d'identification fournies par ce rôle. Si vous ne limitez pas l'accès aux informations d'identification fournies au rôle IAM du nœud Amazon EKS, le Pod a toujours accès à ces informations d'identification. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vos Pods ne peuvent pas interagir avec les services comme prévu, suivez les étapes ci-dessous pour vérifier que tout est correctement configuré.

a. Vérifiez que vos Pods utilisent une version du AWS SDK qui permet d'assumer un rôle IAM via un fichier de jeton d'identité Web OpenID Connect. Pour de plus amples informations, veuillez consulter the section called “Soutenu SDKs”.

b. Vérifiez que le déploiement utilise le compte de service.

```
kubectl describe deployment my-app | grep "Service Account"
```

L'exemple qui suit illustre un résultat.

```
Service Account:  my-service-account
```

c. Si vos Pods ne peuvent toujours pas accéder aux services, consultez les étapes décrites dans Attribuer des rôles IAM aux comptes de service Kubernetes pour vérifier que votre rôle et votre compte de service sont correctement configurés.

Configurer le point de terminaison du service AWS Security Token pour un compte de service

Si vous utilisez un compte de service Kubernetes avec des rôles IAM pour les comptes de service, vous pouvez configurer le type de point de terminaison du service AWS Security Token utilisé par le compte de service si la version de votre cluster et de votre plateforme est identique ou ultérieure à celles répertoriées dans le tableau suivant. Si votre version de Kubernetes ou de votre plateforme est antérieure à celles répertoriées dans le tableau, vos comptes de service ne peuvent utiliser que le point de terminaison global.

Version de Kubernetes	Version de la plateforme	Type de point de terminaison par défaut
`1.31`	`eks.4`	Régional
`1.30`	`eks.2`	Régional
`1.29`	`eks.1`	Régional
`1.28`	`eks.1`	Régional
`1.27`	`eks.1`	Régional
`1.26`	`eks.1`	Régional

AWS recommande d'utiliser les points de terminaison AWS STS régionaux au lieu du point de terminaison global. Cela réduit la latence, fournit une redondance intégrée et augmente la validité des jetons de session. Le service AWS Security Token doit être actif dans la AWS région où le Pod est exécuté. En outre, votre application doit être dotée d'une redondance intégrée pour une autre AWS région en cas de panne du service dans la AWS région. Pour plus d'informations, consultez la section Gestion du AWS STS dans une AWS région dans le guide de l'utilisateur IAM.

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un à l'aide de l'un des guides inclus dansMise en route.
- Un fournisseur IAM OIDC existant pour votre cluster. Pour de plus amples informations, veuillez consulter the section called “Fournisseur IAM OIDC”.

- Un compte de service Kubernetes existant configuré pour être utilisé avec la fonction Amazon EKS IAM pour les comptes de service.

Les exemples suivants utilisent tous le compte de service Kubernetes aws-node utilisé par le plugin Amazon VPC CNI. Vous pouvez les *example values* remplacer par vos propres comptes de service, pods, espaces de noms et autres ressources.

1. Sélectionnez un Pod qui utilise un compte de service dont vous souhaitez modifier le point de terminaison. Déterminez AWS dans quelle région fonctionne le Pod. *aws-node-6mfgv*Remplacez-le par le nom de votre Pod et *kube-system* par l'espace de noms de votre Pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   L'exemple qui suit illustre un résultat.

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   Dans la sortie précédente, le Pod s'exécute sur un nœud de la région us-west-2. AWS

2. Déterminez le type de point de terminaison utilisé par le compte de service du Pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
   ```

   L'exemple qui suit illustre un résultat.

   ```
   AWS_STS_REGIONAL_ENDPOINTS: regional
   ```

   Si le point de terminaison actuel est international, alors `global` est renvoyé en sortie. Si aucune sortie n'est renvoyée, le type de point de terminaison par défaut est en cours d'utilisation et n'a pas été remplacé.

3. Si la version de votre cluster ou celle de votre plateforme est identique ou ultérieure à celles répertoriées dans le tableau, vous pouvez modifier le type de point de terminaison utilisé par votre compte de service pour le faire passer du type par défaut à un type différent à l'aide de l'une des commandes suivantes. Remplacez *aws-node* par le nom de votre compte de service et *kube-system* par l'espace de noms de votre compte de service.

- Si votre type de point de terminaison par défaut ou actuel est international et que vous souhaitez le remplacer par régional :

  ```
  kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
  ```

  Si vous utilisez des rôles IAM pour des comptes de service afin de générer des S3 pré-signés URLs dans votre application exécutée dans les conteneurs de Pods, le format de l'URL des points de terminaison régionaux est similaire à celui de l'exemple suivant :

  ```
  https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
  ```

- Si votre type de point de terminaison par défaut ou actuel est régional et que vous souhaitez le remplacer par international :

  ```
  kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
  ```

  Si votre application envoie explicitement des demandes aux points de terminaison mondiaux AWS STS et que vous ne remplacez pas le comportement par défaut consistant à utiliser des points de terminaison régionaux dans les clusters Amazon EKS, les demandes échoueront avec une erreur. Pour de plus amples informations, veuillez consulter the section called "Les conteneurs Pod reçoivent l'erreur suivante : `An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`".

  Si vous utilisez des rôles IAM pour des comptes de service afin de générer des S3 pré-signés URLs dans votre application exécutée dans les conteneurs de Pods, le format de l'URL des points de terminaison globaux est similaire à celui de l'exemple suivant :

  ```
  https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
  ```

Si vous disposez d'une automatisation qui attend l'URL pré-signée dans un certain format ou si votre application ou les dépendances en aval qui utilisent le pré-signé URLs ont des attentes pour la AWS région ciblée, apportez les modifications nécessaires pour utiliser le point de terminaison AWS STS approprié.

4. Supprimez et recréez tous les pods existants associés au compte de service pour appliquer les variables d'environnement d'identification. Le webhook mutant ne les applique pas aux pods déjà en cours d'exécution. Vous pouvez remplacer *Podskube-system*, et par *-l k8s-app=aws-node* les informations relatives aux pods pour lesquels vous avez défini votre annotation.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Vérifiez que tous les pods ont redémarré.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Affichez les variables d'environnement de l'un des pods. Vérifiez que la valeur `AWS_STS_REGIONAL_ENDPOINTS` est celle que vous avez définie à une étape précédente.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

L'exemple qui suit illustre un résultat.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Authentifiez-vous sur un autre compte auprès de l'IRSA

Vous pouvez configurer les autorisations IAM entre comptes soit en créant un fournisseur d'identité à partir du cluster d'un autre compte, soit en utilisant des opérations en chaîne`AssumeRole`. Dans les exemples suivants, le compte A possède un cluster Amazon EKS qui prend en charge les rôles IAM pour les comptes de service. Les pods qui s'exécutent sur ce cluster doivent bénéficier des autorisations IAM du compte B.

Example Création d'un fournisseur d'identité à partir du cluster d'un autre compte

Example

Dans cet exemple, le Compte A fournit au Compte B l'URL de l'émetteur OpenID Connect (OIDC) à partir de son cluster. Le compte B suit les instructions de la section Créer un fournisseur IAM OIDC pour votre cluster et the section called "Attribuer un rôle IAM" utilise l'URL de l'émetteur OIDC du cluster du compte A. Ensuite, un administrateur de cluster annote le compte de service dans le cluster du compte A pour utiliser le rôle du compte B (*444455556666*).

```
apiVersion: v1
```

```
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::444455556666:role/account-b-role
```

Example Utiliser des opérations **`AssumeRole`** chaînées

Example

Dans cet exemple, le compte B crée une politique IAM avec les autorisations à accorder aux pods du cluster du compte A. Le compte B (*444455556666*) associe cette politique à un rôle IAM doté d'une relation de confiance qui autorise l'`AssumeRole`accès au compte A (*111122223333*).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

Le compte A crée un rôle avec une politique de confiance qui obtient les informations d'identification du fournisseur d'identité créé avec l'adresse de l'émetteur OIDC du cluster.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity"
    }
  ]
```

```
}
```

Si votre cluster se trouve dans une autre , ou si vous avez copié les images dans votre propre référentiel lors d'une étape précédente, effectuez les étapes suivantes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws: iam::444455556666:role/account-b-role"
        }
    ]
}
```

Le code d'application permettant à Pods d'assumer le rôle du compte B utilise deux profils : `account_b_role` et`account_a_role`. Le profil `account_b_role` utilise le profil `account_a_role` comme source. Pour la AWS CLI, le `~/.aws/config` fichier est similaire au suivant.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws: iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws: iam::111122223333:role/account-a-role
```

Pour spécifier des profils chaînés pour d'autres AWS SDKs, consultez la documentation du SDK que vous utilisez. Pour plus d'informations, consultez la section Outils sur lesquels vous pouvez vous appuyer AWS.

Utiliser IRSA avec le SDK AWS

Utilisation des informations d'identification

Pour utiliser les informations d'identification issues des rôles IAM pour les comptes de service (IRSA), votre code peut utiliser n'importe quel AWS SDK pour créer un client pour un AWS service doté d'un SDK. Par défaut, le SDK recherche les informations d' AWS identification Identity and Access Management à utiliser dans une chaîne d'emplacements. Les rôles IAM pour les informations

d'identification des comptes de service seront utilisés si vous ne spécifiez pas de fournisseur d'informations d'identification lorsque vous créez le client ou que vous initialisez le SDK.

Cela fonctionne parce que les rôles IAM des comptes de service ont été ajoutés en tant qu'étape dans la chaîne d'informations d'identification par défaut. Si vos charges de travail utilisent actuellement des informations d'identification qui se trouvent plus haut dans la chaîne d'informations d'identification, ces informations d'identification continueront d'être utilisées même si vous configurez un rôle IAM des comptes de service pour la même charge de travail.

Le SDK échange automatiquement le jeton OIDC du compte de service contre des informations d'identification temporaires provenant du AWS Security Token Service en utilisant l'`AssumeRoleWithWebIdentity`action. Amazon EKS et cette action du kit SDK poursuivent la rotation des informations d'identification temporaires en les renouvelant avant qu'elles n'expirent.

Lorsque vous utilisez des rôles IAM pour des comptes de service, les conteneurs de vos pods doivent utiliser une version du AWS SDK qui permet d'assumer un rôle IAM via un fichier de jeton d'identité Web OpenID Connect. Assurez-vous que vous utilisez les versions suivantes, ou des versions ultérieures, pour votre AWS SDK :

- Java (version 2) – 2.10.11
- Java — 1.12.782
- AWS SDK pour Go v1 — 1.23.13
- AWS SDK pour Go v2 — Toutes les versions sont prises en charge
- Python (Boto3) – 1.9.220
- Python (botocore) – 1.12.200
- AWS CLI — 1.16.232
- Node – 2.525.0 et 3.27.0
- Ruby – 3.58.0
- C++ – 1.7.174
- .NET – 3.3.659.1 – Vous devez également inclure `AWSSDK.SecurityToken`.
- PHP – 3.110.7

De nombreux modules complémentaires Kubernetes populaires, tels que le Cluster Autoscaler, le trafic Internet Route with Load Balancer Controller et le plug-in Amazon VPC CNI pour Kubernetes, prennent en AWS charge les rôles IAM pour les comptes de service.

Pour vous assurer que vous utilisez un SDK compatible, suivez les instructions d'installation de votre SDK préféré sur Tools to Build on AWS lorsque vous créez vos conteneurs.

Considérations

Java

Lorsque vous utilisez Java, vous devez inclure le `sts` module dans le chemin de classe. Pour plus d'informations, consultez WebIdentityTokenFileCredentialsProviderla documentation du SDK Java.

Récupérez les clés de signature pour valider les jetons OIDC

Kubernetes émet un `ProjectedServiceAccountToken` pour chaque compte de service Kubernetes. Ce jeton est un jeton OIDC, qui est en outre un type de jeton Web JSON (JWT). Amazon EKS héberge un point de terminaison OIDC public pour chaque cluster qui contient les clés de signature du jeton afin que les systèmes externes puissent le valider.

Pour valider un`ProjectedServiceAccountToken`, vous devez récupérer les clés de signature publiques OIDC, également appelées ensemble de clés Web JSON (JWKS). Utilisez ces clés dans votre application pour valider le jeton. Par exemple, vous pouvez utiliser la bibliothèque Python PyJWT pour valider des jetons à l'aide de ces clés. Pour plus d'informations sur le`ProjectedServiceAccountToken`, voirthe section called "Informations générales sur IAM, Kubernetes et OpenID Connect (OIDC)".

Prérequis

- Un fournisseur IAM (AWS Identity and Access Management) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".
- AWS CLI : outil de ligne de commande permettant de travailler avec AWS des services, notamment Amazon EKS. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande. Après avoir installé la AWS CLI, nous vous recommandons de la configurer également. Pour plus d'informations, consultez la section Configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.

Procédure

1. Récupérez l'URL OIDC de votre cluster Amazon EKS à l'aide de la AWS CLI.

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. Récupérez la clé de signature publique à l'aide de curl ou d'un outil similaire. Le résultat est un ensemble de clés Web JSON (JWKS).

> **Important**
>
> Amazon EKS limite les appels vers le point de terminaison OIDC. Vous devez mettre en cache la clé de signature publique. Respectez l'`cache-control`en-tête inclus dans la réponse.

> **Important**
>
> Amazon EKS fait pivoter la clé de signature OIDC tous les sept jours.

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}]
```

Découvrez comment EKS Pod Identity permet aux pods d'accéder aux AWS services

Les applications situées dans les conteneurs d'un pod peuvent utiliser un AWS SDK ou la AWS CLI pour envoyer des demandes d'API aux AWS services à l'aide des AWS autorisations Identity and Access Management (IAM). Les applications doivent signer leurs demandes AWS d'API avec des AWS informations d'identification.

EKS Pod Identities permet de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance Amazon fournissent des informations d'identification aux EC2 instances Amazon. Au lieu de créer et de distribuer vos AWS informations d'identification aux conteneurs ou d'utiliser le rôle de l' EC2 instance Amazon, vous associez un rôle IAM à un compte de service Kubernetes et configurez vos Pods pour utiliser le compte de service.

Chaque identité du pod EKS mappe un rôle à un compte de service dans un espace de noms dans le cluster spécifié. Si vous avez la même application dans plusieurs clusters, vous pouvez faire des associations identiques dans chaque cluster sans modifier la politique d'approbation du rôle.

Si un pod utilise un compte de service qui a une association, Amazon EKS définit des variables d'environnement dans les conteneurs du pod. Les variables d'environnement configurent AWS SDKs, y compris la AWS CLI, pour utiliser les informations d'identification EKS Pod Identity.

Avantages des identités du pod EKS

Les identités du pod EKS offrent les avantages suivants :

- Privilège minimal : vous pouvez attribuer des autorisations IAM à un compte de service, et seuls les pods qui utilisent ce compte de service ont accès à ces autorisations. Cette fonctionnalité élimine également le besoin de solutions tierces telles que `kiam` ou `kube2iam`.
- Isolation des informations d'identification : lorsque l'accès à l'Amazon EC2 Instance Metadata Service (IMDS) est restreint, les conteneurs d'un pod peuvent uniquement récupérer les informations d'identification pour le rôle IAM associé au compte de service utilisé par le conteneur. Un conteneur n'a jamais accès aux informations d'identification utilisées par d'autres conteneurs dans d'autres pods. Si l'IMDS n'est pas restreint, les conteneurs du pod ont également accès au rôle IAM du nœud Amazon EKS et les conteneurs peuvent être en mesure d'accéder aux informations d'identification des rôles IAM des autres pods sur le même nœud. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

> **Note**
>
> Les pods configurés avec `hostNetwork: true` auront toujours un accès IMDS, mais la CLI AWS SDKs and utilisera les informations d'identification du Pod Identity lorsqu'elle est activée.

- Auditabilité — La journalisation des accès et des événements est disponible AWS CloudTrail pour faciliter l'audit rétrospectif.

> **Important**
>
> Les conteneurs ne constituent pas une limite de sécurité, et l'utilisation de Pod Identity n'y change rien. Les pods assignés au même nœud partageront un noyau et potentiellement d'autres ressources en fonction de la configuration de votre pod. Alors que les pods exécutés sur des nœuds distincts seront isolés au niveau de la couche de calcul, certaines applications de nœuds disposent d'autorisations supplémentaires dans l'API Kubernetes au-delà du cadre d'une instance individuelle. Par exemple `kubeletkube-proxy`, les pilotes de stockage CSI ou vos propres applications Kubernetes.

EKS Pod Identity est une méthode plus simplethe section called "Identifiants auprès de l'IRSA", car elle n'utilise pas de fournisseurs d'identité OIDC. L'identité du pod EKS présente les améliorations suivantes :

- Opérations indépendantes — Dans de nombreuses organisations, la création de fournisseurs d'identité OIDC relève de la responsabilité d'équipes différentes de celle de l'administration des clusters Kubernetes. L'identité du pod EKS présente une séparation nette des tâches, où toute la configuration des associations de l'identité du pod EKS est effectuée dans Amazon EKS et toute la configuration des autorisations IAM est effectuée dans IAM.
- Réutilisabilité : l'identité du pod EKS utilise un seul principal IAM au lieu des principaux distincts pour chaque cluster que les rôles IAM des comptes de service utilisent. Votre administrateur IAM ajoute le principal suivant à la politique d'approbation de n'importe quel rôle pour le rendre utilisable par les identités du pod EKS.

  ```
  "Principal": {
      "Service": "pods.eks.amazonaws.com"
  }
  ```

- Évolutivité — Chaque ensemble d'informations d'identification temporaires est utilisé par le service EKS Auth dans EKS Pod Identity, et non par chaque AWS SDK que vous exécutez dans chaque module. Ensuite, l'agent d'identité Amazon EKS Pod qui s'exécute sur chaque nœud émet les informations d'identification au SDKs. Ainsi, la charge est réduite à une fois pour chaque nœud et n'est pas dupliquée dans chaque pod. Pour plus de détails sur le processus, consultez the section called "Comment ça marche".

Pour plus d'informations sur la comparaison des deux solutions, consultez the section called "Accès à la charge de travail à AWS ".

Présentation de la configuration des identités du pod EKS

Activez les identités du pod EKS en effectuant les procédures suivantes :

1. the section called "Configuration de l'agent"— Vous n'effectuez cette procédure qu'une seule fois pour chaque cluster. Il n'est pas nécessaire de terminer cette étape si le mode automatique EKS est activé sur votre cluster.
2. the section called "Attribuer un rôle IAM"— Effectuez cette procédure pour chaque ensemble unique d'autorisations que vous souhaitez attribuer à une application.
3. the section called "Compte de service Pod"— Complétez cette procédure pour chaque Pod qui a besoin d'accéder aux AWS services.
4. the section called "Soutenu SDKs"— Vérifiez que la charge de travail utilise un AWS SDK d'une version prise en charge et qu'elle utilise la chaîne d'informations d'identification par défaut.

Limites

- Jusqu'à 5 000 associations d'identité EKS Pod par cluster pour mapper les rôles IAM aux comptes de service Kubernetes sont prises en charge.

Considérations

- Association de rôles IAM : chaque compte de service Kubernetes d'un cluster peut être associé à un rôle IAM à partir du même compte que AWS le cluster. Pour modifier le rôle, modifiez l'association EKS Pod Identity. Pour l'accès entre comptes, déléguez l'accès au rôle à l'aide des rôles IAM. Pour en savoir plus, consultez la section Déléguer l'accès entre les AWS comptes à l'aide de rôles IAM dans le Guide de l'utilisateur IAM.
- EKS Pod Identity Agent : L'agent Pod Identity est requis pour utiliser EKS Pod Identity. L'agent s'exécute en tant que Kubernetes `DaemonSet` sur les nœuds du cluster, fournissant des informations d'identification uniquement aux pods du même nœud. Il utilise le port d'occupation du `hostNetwork` nœud `80` et `2703` l'adresse locale du lien (`169.254.170.23`pour IPv4, `[fd00:ec2::23]` pour IPv6). S'il IPv6 est désactivé dans votre cluster, désactivez-le IPv6 pour le Pod Identity Agent. Pour en savoir plus, voir Désactiver IPv6 dans l'agent d'identité EKS Pod.

- Cohérence éventuelle : les associations EKS Pod Identity sont finalement cohérentes, avec des délais potentiels de plusieurs secondes après les appels d'API. Évitez de créer ou de mettre à jour des associations dans des chemins de code critiques à haute disponibilité. Effectuez plutôt ces actions dans le cadre de routines d'initialisation ou de configuration distinctes et moins fréquentes. Pour en savoir plus, consultez la section Groupes de sécurité par pod dans le guide des meilleures pratiques d'EKS.
- Considérations relatives au proxy et au groupe de sécurité : pour les pods utilisant un proxy, ajoutez `169.254.170.23` (IPv4) et `[fd00:ec2::23]` (IPv6) aux variables d'`no_proxy/NO_PROXY`environnement afin d'éviter l'échec des demandes adressées à l'agent EKS Pod Identity. Si vous utilisez des groupes de sécurité pour les pods avec le AWS VPC CNI, définissez l'`ENABLE_POD_ENI`indicateur sur « true » et le `POD_SECURITY_GROUP_ENFORCING_MODE` drapeau sur « standard ». Pour en savoir plus, voir Attribuer des groupes de sécurité à des pods individuels.

Versions du cluster de l'identité du pod EKS

Pour utiliser EKS Pod Identity, le cluster doit disposer d'une version de plate-forme identique ou ultérieure à la version répertoriée dans le tableau suivant, ou d'une version de Kubernetes ultérieure aux versions répertoriées dans le tableau. Pour trouver la version suggérée de l'agent d'identité Amazon EKS Pod pour une version de Kubernetes, consultez. the section called "Vérifiez la compatibilité"

Version de Kubernetes	Version de la plateforme
Versions de Kubernetes non répertoriées	Toutes les versions de plateforme sont prises en charge
`1.28`	`eks.4`
`1.27`	`eks.8`
`1.26`	`eks.9`

Restrictions relatives à l'identité du pod EKS

Les identités du pod EKS sont disponibles sur les éléments suivants :

- Les versions du cluster Amazon EKS répertoriées dans la rubrique précédente the section called "Versions du cluster de l'identité du pod EKS".
- Nœuds de travail du cluster qui sont des EC2 instances Amazon Linux.

Les identités EKS Pod ne sont pas disponibles dans les pays suivants :

- AWS Outposts.
- Amazon EKS Anywhere.
- Clusters Kubernetes que vous créez et exécutez sur Amazon. EC2 Les composants de l'identité du pod Amazon EKS sont uniquement disponibles sur Amazon EKS.

Vous ne pouvez pas utiliser EKS Pod Identities avec :

- Des pods qui s'exécutent n'importe où, sauf sur EC2 les instances Linux Amazon. Les pods Linux et Windows qui s'exécutent sur AWS Fargate (Fargate) ne sont pas pris en charge. Les pods qui s'exécutent sur des EC2 instances Amazon Windows ne sont pas pris en charge.

Découvrez le fonctionnement d'EKS Pod Identity

Les associations Amazon EKS Pod Identity permettent de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance Amazon fournissent des informations d'identification aux EC2 instances Amazon.

L'identité du pod Amazon EKS fournit des informations d'identification à vos charges de travail avec une API d'authentification EKS supplémentaire et un pod d'agent qui s'exécute sur chaque nœud.

Dans vos modules complémentaires, tels que les modules complémentaires Amazon EKS, le contrôleur autogéré, les opérateurs et les autres modules complémentaires, l'auteur doit mettre à jour son logiciel pour utiliser la dernière version AWS SDKs. Pour la liste de compatibilité entre l'identité du pod EKS et les modules complémentaires produits par Amazon EKS, reportez-vous à la section précédente the section called "Restrictions relatives à l'identité du pod EKS".

Utilisation des identités du pod EKS dans votre code

Dans votre code, vous pouvez utiliser le AWS SDKs pour accéder aux AWS services. Vous écrivez du code pour créer un client pour un AWS service avec un SDK, et par défaut, le SDK recherche les informations d'identification AWS Identity and Access Management à utiliser dans une chaîne d'emplacements. Une fois les informations d'identification valides trouvées, la recherche

s'arrête. Pour plus d'informations sur les emplacements par défaut utilisés, consultez la chaîne de fournisseurs d'informations d'identification dans le guide de référence AWS SDKs et Tools.

Les identités du pod EKS ont été ajoutées au fournisseur d'informations d'identification du conteneur qui est recherché dans une étape de la chaîne d'informations d'identification par défaut. Si vos charges de travail utilisent actuellement des informations d'identification antérieures dans la chaîne de fournisseurs d'informations d'identification, ces informations continueront d'être utilisées même si vous configurez une association d'identité du pod EKS pour la même charge de travail. Ainsi, vous pouvez migrer en toute sécurité à partir d'autres types d'informations d'identification en créant d'abord l'association, avant de supprimer les anciennes informations d'identification.

Le fournisseur d'informations d'identification de conteneur fournit des informations d'identification temporaires à partir d'un agent qui s'exécute sur chaque nœud. Dans Amazon EKS, l'agent est l'agent d'identité du pod Amazon EKS et sur Amazon Elastic Container Service, l'agent est le `amazon-ecs-agent`. Ils SDKs utilisent des variables d'environnement pour localiser l'agent auquel se connecter.

En revanche, les rôles IAM pour les comptes de service fournissent un jeton d'identité Web que le AWS SDK doit utiliser pour échanger avec le AWS Security Token Service. `AssumeRoleWithWebIdentity`

Comment fonctionne EKS Pod Identity Agent avec un Pod

1. Lorsqu'Amazon EKS démarre un nouveau pod qui utilise un compte de service associé à une association EKS Pod Identity, le cluster ajoute le contenu suivant au manifeste du pod :

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
```

```
            audience: pods.eks.amazonaws.com
            expirationSeconds: 86400 # 24 hours
            path: eks-pod-identity-token
```

2. Kubernetes sélectionne le nœud sur lequel exécuter le pod. Ensuite, l'agent Amazon EKS Pod Identity sur le nœud utilise l'AssumeRoleForPodIdentityaction pour récupérer des informations d'identification temporaires à partir de l'API EKS Auth.
3. L'agent d'identité EKS Pod met ces informations d'identification à la disposition de ceux AWS SDKs que vous exécutez dans vos conteneurs.
4. Vous utilisez le kit SDK dans votre application sans spécifier de fournisseur d'informations d'identification pour utiliser la chaîne d'informations d'identification par défaut. Vous pouvez également spécifier le fournisseur d'informations d'identification du conteneur. Pour plus d'informations sur les emplacements par défaut utilisés, consultez la chaîne de fournisseurs d'informations d'identification dans le guide de référence AWS SDKs et Tools.
5. Le kit SDK utilise les variables d'environnement pour se connecter à l'agent d'identité du pod EKS et récupérer les informations d'identification.

> **Note**
>
> Si vos charges de travail utilisent actuellement des informations d'identification antérieures dans la chaîne de fournisseurs d'informations d'identification, ces informations continueront d'être utilisées même si vous configurez une association d'identité du pod EKS pour la même charge de travail.

Configuration de l'agent d'identité Amazon EKS Pod

Les associations Amazon EKS Pod Identity permettent de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance Amazon fournissent des informations d'identification aux EC2 instances Amazon.

L'identité du pod Amazon EKS fournit des informations d'identification à vos charges de travail avec une API d'authentification EKS supplémentaire et un pod d'agent qui s'exécute sur chaque nœud.

> **Tip**
>
> Il n'est pas nécessaire d'installer l'agent EKS Pod Identity sur les clusters en mode automatique EKS. Cette fonctionnalité est intégrée au mode automatique d'EKS.

Considérations

- Par défaut, l'agent EKS Pod Identity écoute sur une `IPv6` adresse `IPv4` et les pods pour demander des informations d'identification. L'agent utilise l'adresse IP de bouclage (localhost) `169.254.170.23` pour `IPv4` et l'adresse IP localhost pour. `[fd00:ec2::23]` `IPv6`
- Si vous désactivez `IPv6` des adresses ou si vous empêchez les adresses `IPv6` IP de l'hôte local, l'agent ne peut pas démarrer. Pour démarrer l'agent sur des nœuds qui ne peuvent pas être utilisés`IPv6`, suivez les étapes décrites the section called "Désactiver IPv6" pour désactiver la `IPv6` configuration.

Création d'un agent d'identité du pod Amazon EKS

Conditions préalables pour l'agent

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route. La version du cluster et la version de la plate-forme doivent être identiques ou ultérieures aux versions répertoriées dans les versions du cluster EKS Pod Identity.
- Le rôle de nœud dispose d'autorisations permettant à l'agent d'effectuer l'action `AssumeRoleForPodIdentity` dans l'API d'authentification EKS. Vous pouvez utiliser la politique AWS gérée : Amazon EKSWorker NodePolicy ou ajouter une politique personnalisée similaire à la suivante :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

Cette action peut être limitée par des balises pour restreindre les rôles qui peuvent être assumés par les pods qui utilisent l'agent.

- Les nœuds peuvent atteindre et télécharger des images à partir d'Amazon ECR. L'image de conteneur pour le module complémentaire se trouve dans les registres répertoriés dans Afficher les registres d'images de conteneur Amazon pour les modules complémentaires Amazon EKS.

  Notez que vous pouvez modifier l'emplacement de l'image et `imagePullSecrets` prévoir des modules complémentaires EKS dans les paramètres de configuration facultatifs de et `--configuration-values` dans la AWS CLI. AWS Management Console
- Les nœuds peuvent accéder à l'API d'authentification Amazon EKS. Pour les clusters privés, le `eks-auth` point de terminaison AWS PrivateLink est obligatoire.

Agent d'installation avec AWS console

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire l'agent d'identité du pod EKS.
3. Choisissez l'onglet Modules complémentaires.
4. Choisissez Obtenez plus de modules complémentaires.
5. Cochez la case en haut à droite du module complémentaire de l'agent d'identité du pod EKS, puis sélectionnez Suivant.
6. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez n'importe quelle Version dans la liste déroulante Version.
7. (Facultatif) Développez les Paramètres de configuration facultatifs pour entrer une configuration supplémentaire. Par exemple, vous pouvez fournir un autre emplacement d'image de conteneur et `ImagePullSecrets`. Le schéma JSON avec les clés acceptées est affiché dans Schéma de configuration du module complémentaire.

   Saisissez les clés et les valeurs de configuration dans Valeurs de configuration.
8. Choisissez Suivant.
9. Confirmez que les pods de l'agent d'identité du pod EKS sont en cours d'exécution sur votre cluster.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

   L'exemple qui suit illustre un résultat.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Vous pouvez maintenant utiliser les associations d'identité du pod EKS dans votre cluster. Pour de plus amples informations, veuillez consulter the section called "Attribuer un rôle IAM".

Agent de configuration avec AWS CLI

1. Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --
addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> L'agent d'identité EKS Pod n'utilise pas les rôles *`service-account-role-arn`* for IAM pour les comptes de service. Vous devez fournir à l'agent d'identité du pod EKS des autorisations dans le rôle de nœud.

2. Confirmez que les pods de l'agent d'identité du pod EKS sont en cours d'exécution sur votre cluster.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

L'exemple qui suit illustre un résultat.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Vous pouvez maintenant utiliser les associations d'identité du pod EKS dans votre cluster. Pour de plus amples informations, veuillez consulter the section called "Attribuer un rôle IAM".

Attribuer un rôle IAM à un compte de service Kubernetes

Cette rubrique explique comment configurer un compte de service Kubernetes pour qu'il assume un rôle de gestion des AWS identités et des accès (IAM) avec EKS Pod Identity. Tous les pods configurés pour utiliser le compte de service peuvent ensuite accéder à tout AWS service auquel le rôle est autorisé à accéder.

Pour créer une association EKS Pod Identity, il n'y a qu'une seule étape : vous créez l'association dans EKS via la AWS Management Console AWS CLI AWS CloudFormation et d'autres outils. AWS SDKs Il n'existe aucune donnée ou métadonnée concernant les associations au sein du cluster dans les objets Kubernetes et vous n'ajoutez aucune annotation aux comptes de service.

Prérequis

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un en suivant l'un des guides présentés dansMise en route.
- Le principal IAM qui crée l'association doit avoir le `iam:PassRole`.
- La dernière version de la AWS CLI installée et configurée sur votre appareil ou AWS CloudShell. Vous pouvez vérifier votre version actuelle avec `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée dans le AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Cluster d'accès avec kubectl".

Création d'une association Pod Identity (AWS console)

1. Ouvrez la console Amazon EKS.

2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire l'agent d'identité du pod EKS.
3. Choisissez l'onglet Access.
4. Dans Associations d'identité du pod, sélectionnez Créer.
5. Pour Rôle IAM, sélectionnez le rôle IAM avec les autorisations dont a besoin la charge de travail.

> **Note**
>
> La liste ne contient que les rôles qui ont la politique d'approbation suivante qui permet à l'identité du pod EKS de les utiliser.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`— EKS Pod `AssumeRole` Identity assume le rôle IAM avant de transmettre les informations d'identification temporaires à vos pods.

`sts:TagSession`— EKS Pod Identity est utilisé `TagSession` pour inclure des balises de session dans les demandes adressées à AWS STS.

Vous pouvez utiliser ces balises dans les clés de condition de la politique de confiance pour limiter les comptes de service, les espaces de noms et les clusters autorisés à utiliser ce rôle.

Pour obtenir la liste des clés de condition Amazon EKS, consultez la rubrique Conditions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

6. Pour l'espace de noms Kubernetes, sélectionnez l'espace de noms Kubernetes qui contient le compte de service et la charge de travail. Vous pouvez éventuellement spécifier un espace de noms par un nom qui n'existe pas dans le cluster.
7. Pour le compte de service Kubernetes, sélectionnez le compte de service Kubernetes à utiliser. Le manifeste de votre charge de travail Kubernetes doit spécifier ce compte de service. Vous pouvez éventuellement spécifier un compte de service par un nom qui n'existe pas dans le cluster.
8. (Facultatif) Pour Balises, choisissez Ajouter une balise pour ajouter des métadonnées dans une paire clé et valeur. Ces balises sont appliquées à l'association et peuvent être utilisées dans les politiques IAM.

   Vous pouvez répéter cette étape pour ajouter plusieurs balises.
9. Sélectionnez Create (Créer).

Création d'une association Pod Identity (AWS CLI)

1. Si vous souhaitez associer une politique IAM existante à votre rôle IAM, passez à l'étape suivante.

   Créez une politique IAM. Vous pouvez créer votre propre politique ou copier une politique AWS gérée qui accorde déjà certaines des autorisations dont vous avez besoin et la personnaliser en fonction de vos besoins spécifiques. Pour plus d'informations, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

   a. Créez un fichier qui inclut les autorisations pour les AWS services auxquels vous souhaitez que vos pods accèdent. Pour obtenir la liste de toutes les actions pour tous les AWS services, consultez la référence d'autorisation des services.

      Vous pouvez exécuter la commande suivante pour créer un exemple de fichier de politique autorisant l'accès en lecture seule à un compartiment Amazon S3. Vous pouvez éventuellement stocker des informations de configuration ou un script de démarrage dans ce compartiment, et les conteneurs de votre Pod peuvent lire le fichier depuis le compartiment et le charger dans votre application. Si vous souhaitez créer cet exemple de politique, copiez le contenu suivant sur votre appareil. `my-pod-secrets-bucket`Remplacez-le par le nom de votre compartiment et exécutez la commande.

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

b. Créez la politique IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

2. Créez un rôle IAM et associez-le à un compte de service Kubernetes.

   a. Si vous possédez déjà un compte de service Kubernetes et que vous souhaitez assumer un rôle IAM, vous pouvez ignorer cette étape.

      Créez un compte de service Kubernetes. Copiez les contenus suivants sur votre appareil. *my-service-account*Remplacez-le par le nom de votre choix et *default* par un autre espace de noms, si nécessaire. Si vous modifiez*default*, l'espace de noms doit déjà exister.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

      Exécutez la commande suivante.

```
kubectl apply -f my-service-account.yaml
```

b. Exécutez la commande suivante pour créer un fichier de stratégie d'approbation pour le rôle IAM.

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

c. Créez le rôle. Remplacez *my-role* avec un nom pour votre rôle IAM, et *my-role-description* avec une description de votre rôle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file:// 
trust-relationship.json --description "my-role-description"
```

d. Liez une politique IAM à votre rôle. Remplacez *my-role* par le nom de votre rôle IAM et *my-policy* par le nom d'une politique existante que vous avez créée.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:
 iam::111122223333:policy/my-policy
```

> **Note**
>
> Contrairement aux rôles IAM pour les comptes de service, EKS Pod Identity n'utilise pas d'annotation sur le compte de service.

e. Exécutez la commande suivante pour créer l'association. `my-cluster`Remplacez-le par le nom du cluster, remplacez-le *`my-service-account`* par le nom de votre choix et *`default`* par un autre espace de noms, si nécessaire.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-arn
 arn:aws: iam::111122223333:role/my-role --namespace default --service-account my-
service-account
```

L'exemple qui suit illustre un résultat.

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws: iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> Vous pouvez spécifier un espace de noms et un compte de service par un nom qui n'existe pas dans le cluster. Vous devez créer l'espace de noms, le compte de service et la charge de travail qui utilise le compte de service pour que l'association de l'identité du pod EKS fonctionne.

Confirmer la configuration

1. Vérifiez que la politique de confiance du rôle IAM est correctement configurée.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

L'exemple qui suit illustre un résultat.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow EKS Auth service to assume this role for Pod Identities",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

2. Vérifiez que la stratégie que vous avez associée à votre rôle lors d'une étape précédente est associée au rôle.

```
aws iam list-attached-role-policies --role-name my-role --query
 'AttachedPolicies[].PolicyArn' --output text
```

L'exemple qui suit illustre un résultat.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Définissez une variable pour stocker l'Amazon Resource Name (ARN) de la stratégie que vous souhaitez utiliser. *my-policy*Remplacez-le par le nom de la politique pour laquelle vous souhaitez confirmer les autorisations.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Affichez la version par défaut de la stratégie.

```
aws iam get-policy --policy-arn $policy_arn
```

L'exemple qui suit illustre un résultat.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Consultez le contenu de la politique pour vous assurer qu'elle inclut toutes les autorisations dont votre Pod a besoin. Si nécessaire, remplacez *1* la commande suivante par la version renvoyée dans la sortie précédente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

L'exemple qui suit illustre un résultat.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws: s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Si vous avez créé l'exemple de stratégie lors d'une étape précédente, le résultat est le même. Si vous avez créé une politique différente, le *example* contenu est différent.

Étapes suivantes

the section called "Compte de service Pod"

Accédez aux AWS ressources à l'aide des rôles IAM EKS Pod Identity Target

Lorsque vous exécutez des applications sur Amazon Elastic Kubernetes Service (Amazon EKS), il se peut que vous deviez accéder AWS à des ressources qui existent dans différents comptes. AWS Ce guide explique comment configurer l'accès entre comptes à l'aide d'EKS Pod Identity, qui permet à vos pods Kubernetes d'accéder à d'autres AWS ressources à l'aide de rôles cibles.

Prérequis

Avant de commencer, assurez-vous d'avoir effectué les étapes suivantes :

- Configuration de l'agent d'identité Amazon EKS Pod
- Création d'un rôle EKS Pod Identity

Comment ça marche

Pod Identity permet aux applications de votre cluster EKS d'accéder aux AWS ressources de différents comptes par le biais d'un processus appelé chaînage des rôles.

Lorsque vous créez une association Pod Identity, vous pouvez fournir deux rôles IAM : un rôle EKS Pod Identity dans le même compte que votre cluster EKS et un rôle IAM cible provenant du compte contenant les AWS ressources auxquelles vous souhaitez accéder (comme les compartiments S3 ou les bases de données RDS). Le rôle EKS Pod Identity doit figurer dans le compte de votre cluster EKS en raison des PassRole exigences IAM, tandis que le rôle IAM cible peut se trouver dans n'importe quel AWS compte. PassRole permet à une AWS entité de déléguer l'attribution de rôles à un autre service. EKS Pod Identity PassRole permet de connecter un rôle à un compte de service Kubernetes, ce qui nécessite que le rôle et l'identité qui le transmettent se trouvent dans le même AWS compte que le cluster EKS. Lorsque votre pod d'application a besoin d'accéder à AWS des ressources, il demande des informations d'identification à Pod Identity. Pod Identity exécute ensuite automatiquement deux hypothèses de rôle en séquence : d'abord en assumant le rôle EKS Pod Identity, puis en utilisant ces informations d'identification pour assumer le rôle IAM cible. Ce processus fournit à votre pod des informations d'identification temporaires dotées des autorisations définies dans le rôle cible, ce qui permet un accès sécurisé aux ressources d'autres AWS comptes.

Considérations concernant la mise en cache

En raison des mécanismes de mise en cache, les mises à jour d'un rôle IAM dans une association Pod Identity existante peuvent ne pas prendre effet immédiatement dans les pods exécutés sur votre cluster EKS. L'agent Pod Identity met en cache les informations d'identification IAM en fonction de

la configuration de l'association au moment où les informations d'identification sont récupérées. Si l'association inclut uniquement un rôle EKS Pod Identity et aucun rôle IAM cible, les informations d'identification mises en cache durent 6 heures. Si l'association inclut à la fois l'ARN du rôle EKS Pod Identity et un rôle Target IAM, les informations d'identification mises en cache durent 59 minutes. La modification d'une association existante, telle que la mise à jour de l'ARN du rôle EKS Pod Identity ou l'ajout d'un rôle IAM cible, ne réinitialise pas le cache existant. Par conséquent, l'agent ne reconnaîtra pas les mises à jour tant que les informations d'identification mises en cache ne seront pas actualisées. Pour appliquer les modifications plus rapidement, vous pouvez recréer les modules existants ; sinon, vous devrez attendre que le cache expire.

Étape 1 : créer et associer un rôle IAM cible

Au cours de cette étape, vous allez établir une chaîne de confiance sécurisée en créant et en configurant un rôle IAM cible. À titre de démonstration, nous allons créer un nouveau rôle IAM cible pour établir une chaîne de confiance entre deux AWS comptes : le rôle EKS Pod Identity (par exemple`eks-pod-identity-primary-role`) du AWS compte du cluster EKS est autorisé à assumer le rôle IAM cible (par exemple`eks-pod-identity-aws-resources`) sur votre compte cible, ce qui permet d'accéder à des AWS ressources telles que les compartiments Amazon S3.

Création du rôle IAM cible

1. Ouvrez la console Amazon IAM.
2. Dans la barre de navigation supérieure, vérifiez que vous êtes connecté au compte contenant les AWS ressources (telles que des compartiments S3 ou des tables DynamoDB) pour votre rôle IAM cible.
3. Dans le volet de navigation de gauche, choisissez Rôles.
4. Cliquez sur le bouton Créer un rôle, puis sur le AWS compte sous « Type d'entité de confiance ».
5. Choisissez Un autre AWS compte, entrez votre numéro de AWS compte (le compte sur lequel se trouve votre rôle EKS Pod Identity), puis choisissez Next.
6. Ajoutez les politiques d'autorisation que vous souhaitez associer au rôle (par exemple, AmazonS3FullAccess), puis choisissez Next.
7. Entrez un nom de rôle, par exemple`MyCustomIAMTargetRole`, puis choisissez Créer un rôle.

Mettre à jour la politique de confiance du rôle IAM cible

1. Après avoir créé le rôle, vous serez renvoyé à la liste des rôles. Recherchez et sélectionnez le nouveau rôle que vous avez créé à l'étape précédente (par exemple,`MyCustomIAMTargetRole`).

2. Sélectionnez l'onglet Trust Relationships (Relations d'approbation).
3. Cliquez sur Modifier la politique de confiance sur le côté droit.
4. Dans l'éditeur de stratégie, remplacez le JSON par défaut par votre politique de confiance. Remplacez les valeurs d'espace réservé pour le nom du rôle et `111122223333` dans l'ARN du rôle IAM par l'ID du AWS compte hébergeant votre cluster EKS. Vous pouvez également éventuellement utiliser la politique de confiance PrincipalTags dans les rôles pour autoriser uniquement des comptes de service spécifiques d'un cluster et d'un espace de noms donnés à assumer votre rôle cible. Par exemple :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks-cluster-arn": "arn:aws:eks:us-
east-1:111122223333:cluster/example-cluster",
          "aws:PrincipalTag/kubernetes-namespace": "ExampleNameSpace",
          "aws:PrincipalTag/kubernetes-service-account": "ExampleServiceAccountName"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws: iam::111122223333:role/eks-pod-identity-
primary-role"
        }
      }
    }
  ]
}
```

La politique ci-dessus permet au rôle `eks-pod-identeity-primary-role` du AWS compte 111122223333 doté des tags de session EKS Pod Identity appropriés d'assumer ce rôle.

Si vous avez désactivé les balises de session dans votre EKS Pod Identity, EKS Pod Identity définit également les `sts:ExternalId` informations relatives au cluster, à l'espace de noms et au compte de service d'un pod lorsqu'il assume un rôle cible.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "region/111122223333/cluster-name/namespace/service-
account-name"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws: iam::111122223333:role/eks-pod-identity-
primary-role"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws: iam::111122223333:root"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

La politique ci-dessus permet de garantir que seuls le cluster, l'espace de noms et le compte de service attendus peuvent assumer le rôle cible.

Mettre à jour la politique d'autorisation pour le rôle EKS Pod Identity

Au cours de cette étape, vous allez mettre à jour la politique d'autorisation du rôle EKS Pod Identity associé à votre cluster Amazon EKS en ajoutant l'ARN du rôle IAM cible en tant que ressource.

1. Ouvrez la console Amazon EKS.

2. Dans le volet de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom de votre cluster EKS.
3. Choisissez l'onglet Access.
4. Sous Associations Pod Identity, sélectionnez votre rôle EKS Pod Identity.
5. Choisissez Autorisations, Ajouter des autorisations, puis Créer une politique intégrée.
6. Choisissez JSON sur le côté droit.
7. Dans l'éditeur de politique, remplacez le JSON par défaut par votre politique d'autorisation. Remplacez la valeur de l'espace réservé pour le nom du rôle et `222233334444` dans l'ARN du rôle IAM par votre rôle IAM cible. Par exemple :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws: iam::222233334444:role/eks-pod-identity-aws-
resources"
        }
    ]
}
```

Étape 2 : associer le rôle IAM cible à un compte de service Kubernetes

Au cours de cette étape, vous allez créer une association entre le rôle Target IAM et le compte de service Kubernetes dans votre cluster EKS.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster auquel vous souhaitez ajouter l'association.
3. Choisissez l'onglet Access.
4. Dans Associations d'identité du pod, sélectionnez Créer.
5. Choisissez le rôle EKS Pod Identity dans le rôle IAM que vos charges de travail devront assumer.

6. Choisissez le rôle Target IAM dans le rôle Target IAM qui sera assumé par le rôle EKS Pod Identity.
7. Dans le champ espace de noms Kubernetes, entrez le nom de l'espace de noms dans lequel vous souhaitez créer l'association (par exemple,). `my-app-namespace` Cela définit l'emplacement du compte de service.
8. Dans le champ Compte de service Kubernetes, entrez le nom du compte de service (par exemple,`my-service-account`) qui utilisera les informations d'identification IAM. Cela lie le rôle IAM au compte de service.
9. Choisissez Create pour créer l'association.

Configurer les pods pour accéder aux AWS services avec des comptes de service

Si un Pod doit accéder à AWS des services, vous devez le configurer pour utiliser un compte de service Kubernetes. Le compte de service doit être associé à un rôle AWS Identity and Access Management (IAM) autorisé à accéder aux services. AWS

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un à l'aide de l'un des guides inclus dansMise en route.
- Un compte de service Kubernetes existant et une association EKS Pod Identity qui associe le compte de service à un rôle IAM. Le rôle doit être associé à une politique IAM contenant les autorisations dont vous souhaitez que vos pods disposent pour utiliser les AWS services. Pour plus d'informations sur la façon de créer et de configurer le compte de service et le rôle, consultez the section called "Attribuer un rôle IAM".
- La dernière version de la AWS CLI installée et configurée sur votre appareil ou AWS CloudShell. Vous pouvez vérifier votre version actuelle avec `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée dans le AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous

pouvez utiliser la version `kubectl 1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Cluster d'accès avec kubectl".

1. Utilisez la commande suivante pour créer un manifeste de déploiement avec lequel vous pouvez déployer un Pod pour confirmer la configuration. Remplacez les *`example values`* par vos propres valeurs.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Appliquez le manifeste à votre cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Vérifiez que les variables d'environnement requises existent pour votre Pod.

   a. Affichez les pods qui ont été déployés lors du déploiement à l'étape précédente.

```
kubectl get pods | grep my-app
```

   L'exemple qui suit illustre un résultat.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. Vérifiez que le Pod dispose d'un fichier de jetons de compte de service.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

L'exemple qui suit illustre un résultat.

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. Vérifiez que vos pods peuvent interagir avec les AWS services à l'aide des autorisations que vous avez attribuées dans la politique IAM associée à votre rôle.

> **Note**
>
> Lorsqu'un pod utilise les AWS informations d'identification d'un rôle IAM associé à un compte de service, la AWS CLI ou un autre SDKs élément des conteneurs de ce pod utilisent les informations d'identification fournies par ce rôle. Si vous ne limitez pas l'accès aux informations d'identification fournies au rôle IAM du nœud Amazon EKS, le Pod a toujours accès à ces informations d'identification. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vos Pods ne peuvent pas interagir avec les services comme prévu, suivez les étapes ci-dessous pour vérifier que tout est correctement configuré.

a. Vérifiez que vos pods utilisent une version du AWS SDK qui permet d'assumer un rôle IAM via une association EKS Pod Identity. Pour de plus amples informations, veuillez consulter the section called “Soutenu SDKs”.

b. Vérifiez que le déploiement utilise le compte de service.

```
kubectl describe deployment my-app | grep "Service Account"
```

L'exemple qui suit illustre un résultat.

```
Service Account:  my-service-account
```

Accordez aux Pods l'accès aux AWS ressources en fonction des balises

Le contrôle d'accès basé sur les attributs (ABAC) accorde des droits aux utilisateurs par le biais de politiques qui combinent les attributs entre eux. EKS Pod Identity attache des balises aux informations d'identification temporaires de chaque pod avec des attributs tels que le nom du cluster, l'espace de noms et le nom du compte de service. Ces balises de session de rôle permettent aux administrateurs de créer un rôle unique qui peut fonctionner avec tous les comptes de service en autorisant l'accès aux AWS ressources en fonction des balises correspondantes. En ajoutant la prise en charge des balises de session de rôle, vous pouvez appliquer des limites de sécurité plus strictes entre les clusters et les charges de travail au sein des clusters, tout en réutilisant les mêmes rôles IAM et les mêmes politiques IAM.

Exemple de politique avec balises

Vous trouverez ci-dessous un exemple de politique IAM qui accorde des `s3:GetObject` autorisations lorsque l'objet correspondant est étiqueté avec le nom du cluster EKS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
```

```
}
```

Activer ou désactiver les balises de session

EKS Pod Identity ajoute un ensemble prédéfini de balises de session lorsqu'il assume le rôle. Ces balises de session permettent aux administrateurs de créer un rôle unique qui peut fonctionner avec toutes les ressources en autorisant l'accès aux AWS ressources en fonction des balises correspondantes.

Activer les balises de session

Les balises de session sont automatiquement activées avec EKS Pod Identity. Aucune action n'est requise de votre part. Par défaut, EKS Pod Identity associe un ensemble de balises prédéfinies à votre session. Pour référencer ces balises dans les politiques, utilisez la syntaxe `${aws:PrincipalTag/` suivie de la clé de balise. Par exemple, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

Désactiver les balises de session

AWS compresse les politiques de session intégrées, les politiques ARNs gérées et les balises de session dans un format binaire compressé doté d'une limite distincte. Si vous recevez un `PackedPolicyTooLarge` message d'erreur indiquant que le format binaire compressé a dépassé la limite de taille, vous pouvez essayer de réduire la taille en désactivant les balises de session ajoutées par EKS Pod Identity. Pour désactiver ces balises de session, procédez comme suit :

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster que vous souhaitez modifier.
3. Choisissez l'onglet Access.

4. Dans les associations Pod Identity, choisissez l'ID d'association que vous souhaitez modifier dans Association ID, puis choisissez Modifier.
5. Sous Balises de session, choisissez Désactiver les balises de session.
6. Sélectionnez Enregistrer les modifications.

Balises inter-comptes

Toutes les balises de session ajoutées par l'identité du pod EKS sont transitives ; les clés et les valeurs des balises sont transmises à toutes les actions `AssumeRole` que vos charges de travail utilisent pour changer de rôle dans un autre compte. Vous pouvez utiliser ces balises dans les politiques d'autres comptes pour limiter l'accès dans des scénarios intercomptes. Pour plus d'informations, consultez Chaînage des rôles avec des balises de session dans le Guide de l'utilisateur IAM.

Balises personnalisées

EKS Pod Identity ne peut pas ajouter de balises personnalisées supplémentaires à l'`AssumeRole`action qu'il exécute. Toutefois, les balises que vous appliquez au rôle IAM sont toujours disponibles dans le même format : `${aws:PrincipalTag/` suivies de la clé, par exemple`${aws:PrincipalTag/MyCustomTag}`.

> **Note**
>
> Les balises ajoutées à la session via la demande `sts:AssumeRole` sont prioritaires en cas de conflit. Par exemple, disons que :
>
> - Amazon EKS ajoute une clé `eks-cluster-name` et une valeur `my-cluster` à la session lorsqu'EKS assume le rôle de client et
> - Vous ajoutez une `eks-cluster-name` balise au rôle IAM avec la valeur`my-own-cluster`.
>
> Dans ce cas, le premier prévaut et la valeur de la `eks-cluster-name` balise sera`my-cluster`.

Utiliser l'identité du pod avec le AWS SDK

Utilisation des informations d'identification de l'identité du pod EKS

Pour utiliser les informations d'identification issues d'une association EKS Pod Identity, votre code peut utiliser n'importe quel AWS SDK afin de créer un client pour un AWS service doté d'un SDK. Par défaut, le SDK recherche dans une chaîne d'emplacements les informations d'identification AWS Identity and Access Management à utiliser. Les informations d'identification EKS Pod Identity seront utilisées si vous ne spécifiez pas de fournisseur d'informations d'identification lorsque vous créez le client ou que vous initialisez le SDK.

Cela fonctionne parce que les identités du pod EKS ont été ajoutées au fournisseur d'informations d'identification du conteneur qui est recherché dans une étape de la chaîne d'informations d'identification par défaut. Si vos charges de travail utilisent actuellement des informations d'identification antérieures dans la chaîne de fournisseurs d'informations d'identification, ces informations continueront d'être utilisées même si vous configurez une association d'identité du pod EKS pour la même charge de travail.

Pour plus d'informations sur le fonctionnement des identités du pod EKS, consultez the section called "Comment ça marche".

Lorsque vous utilisez Découvrez comment EKS Pod Identity accorde aux pods l'accès aux AWS services, les conteneurs de vos pods doivent utiliser une version du AWS SDK qui permet d'assumer un rôle IAM depuis l'agent EKS Pod Identity. Assurez-vous que vous utilisez les versions suivantes, ou des versions ultérieures, pour votre AWS SDK :

- Java (version 2) : 2.21.30
- Java — 1.12.746
- Go v1 : v1.47.11
- Go v2 : release-2023-11-14
- Python (Boto3) — 1.34.41
- Python (botocore) — 1.34.41
- AWS CLI — 1.30.0

  AWS CLI — 2.15.0
- JavaScript v2 — 2,1550,0
- JavaScript v3 — v3.458.0
- Kotlin — v1.0.1

- Ruby : 3.188.0
- Rust — sortie-2024-03-13
- C++ — 1.11.263
- .NET — 3.7.734.0
- PowerShell — 4,1502
- PHP : 3.287.1

Pour vous assurer que vous utilisez un SDK compatible, suivez les instructions d'installation de votre SDK préféré sur Tools to Build on AWS lorsque vous créez vos conteneurs.

Pour obtenir la liste des modules complémentaires compatibles avec EKS Pod Identity, consultezthe section called "Référence de support Pod Identity".

Désactiver **`IPv6`** dans l'agent d'identité EKS Pod

AWS Management Console

1. Pour le désactiver `IPv6` dans l'agent EKS Pod Identity, ajoutez la configuration suivante aux paramètres de configuration facultatifs du module complémentaire EKS.
   a. Ouvrez la console Amazon EKS.
   b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire.
   c. Choisissez l'onglet Modules complémentaires.
   d. Cochez la case en haut à droite du module complémentaire EKS Pod Identity Agent, puis choisissez Modifier.
   e. Sur la page Configurer l'agent d'identité EKS Pod :
      i. Sélectionnez la version que vous souhaitez utiliser. Nous vous recommandons de conserver la même version que celle de l'étape précédente et de mettre à jour la version et la configuration dans le cadre d'actions distinctes.
      ii. Sélectionnez Paramètres de configuration facultatifs.
      iii. Entrez la clé JSON `"agent":` et la valeur d'un objet JSON imbriqué avec une clé `"additionalArgs":` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`. L'exemple suivant montre que la politique réseau est activée :

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

Cette configuration définit l'`IPv4`adresse comme étant la seule adresse utilisée par l'agent.

f. Pour appliquer la nouvelle configuration en remplaçant les modules EKS Pod Identity Agent, choisissez Enregistrer les modifications.

   Amazon EKS applique les modifications aux modules complémentaires EKS en déployant l'agent Kubernetes `DaemonSet` for EKS Pod Identity. Vous pouvez suivre l'état du déploiement dans l'historique des mises à jour du module complémentaire dans AWS Management Console et avec`kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

   `kubectl rollout`possède les commandes suivantes :

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

   Si le déploiement prend trop de temps, Amazon EKS annulera le déploiement et un message indiquant le type de mise à jour de l'extension et le statut Echec sera ajouté à l'historique des mises à jour de l'extension. Pour étudier les problèmes éventuels, commencez par l'historique du déploiement et exécutez-le `kubectl logs` sur un pod EKS Pod Identity Agent pour consulter les journaux d'EKS Pod Identity Agent.

2. Si la nouvelle entrée dans l'historique des mises à jour a le statut Successful, le déploiement est terminé et le module complémentaire utilise la nouvelle configuration dans tous les pods EKS Pod Identity Agent.

AWS CLI

1. Pour le désactiver `IPv6` dans l'agent EKS Pod Identity, ajoutez la configuration suivante aux valeurs de configuration du module complémentaire EKS.

   Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
       --resolve-conflicts PRESERVE --configuration-values '{"agent":{"additionalArgs":
    { "-b": "169.254.170.23"}}}'
   ```

   Cette configuration définit l'`IPv4`adresse comme étant la seule adresse utilisée par l'agent.

   Amazon EKS applique les modifications aux modules complémentaires EKS en déployant l'agent Kubernetes DaemonSet for EKS Pod Identity. Vous pouvez suivre l'état du déploiement dans l'historique des mises à jour du module complémentaire dans AWS Management Console et avec`kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

   `kubectl rollout`possède les commandes suivantes :

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   undo     -- Undo a previous rollout
   ```

   Si le déploiement prend trop de temps, Amazon EKS annulera le déploiement et un message indiquant le type de mise à jour de l'extension et le statut Echec sera ajouté à l'historique des mises à jour de l'extension. Pour étudier les problèmes éventuels, commencez par l'historique du déploiement et exécutez-le `kubectl logs` sur un pod EKS Pod Identity Agent pour consulter les journaux d'EKS Pod Identity Agent.

Créez un rôle IAM avec la politique de confiance requise par EKS Pod Identity

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

L'identité du pod EKS utilise `AssumeRole` pour assumer le rôle IAM avant de transmettre les informations d'identification temporaires à vos pods.

`sts:TagSession`

EKS Pod `TagSession` Identity inclut des balises de session dans les demandes adressées à AWS STS.

Vous pouvez utiliser ces balises dans les clés de condition de la politique de confiance pour limiter les comptes de service, les espaces de noms et les clusters autorisés à utiliser ce rôle.

Pour obtenir la liste des clés de condition Amazon EKS, consultez la rubrique Conditions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

Gérez les ressources informatiques à l'aide de nœuds

Un nœud Kubernetes est une machine qui exécute des applications conteneurisées. Chaque nœud dispose des composants suivants :

- Container Runtime : logiciel chargé de faire fonctionner les conteneurs.
- kubelet — S'assure que les conteneurs sont sains et fonctionnent dans leur Pod associé.
- kube-proxy — Maintient les règles réseau qui autorisent la communication avec vos pods.

Pour plus d'informations, consultez Nœuds dans la documentation Kubernetes.

Votre cluster Amazon EKS peut planifier des pods sur n'importe quelle combinaison de nœuds gérés en mode automatique EKS, de nœuds autogérés, de groupes de nœuds gérés par Amazon EKS, de AWS Fargate et de nœuds hybrides Amazon EKS. Pour en savoir plus sur les nœuds déployés dans votre cluster, consultez the section called “Accès aux ressources du cluster”.

> **Note**
>
> À l'exception des nœuds hybrides, les nœuds doivent se trouver dans le même VPC que les sous-réseaux que vous avez sélectionnés lors de la création du cluster. Cependant, les nœuds ne doivent pas nécessairement se trouver dans les mêmes sous-réseaux.

Comparez les options de calcul

Le tableau suivant fournit plusieurs critères à évaluer pour décider quelles options répondent le mieux à vos besoins. Les nœuds autogérés sont une autre option qui répond à tous les critères énumérés, mais ils nécessitent beaucoup plus de maintenance manuelle. Pour de plus amples informations, veuillez consulter the section called “Nœuds autogérés”.

> **Note**
>
> Bottlerocket présente quelques différences spécifiques par rapport aux informations générales de ce tableau. Pour plus d'informations, consultez la documentation de Bottlerocket sur. GitHub

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Peut être déployé dans les AWS Outposts	Non	Non	Non
Possibilité d'être déployé dans une zone locale AWS	Oui	Non	Non
Possibilité d'exécuter des conteneurs qui nécessitent Windows	Oui	Non	Non
Possibilité d'exécuter des conteneurs qui nécessitent Linux	Oui	Oui	Oui
Possibilité d'exécuter des applications nécessitant la puce Inferentia	Oui : nœuds Amazon Linux uniquement	Oui	Non
Possibilité d'exécuter des applications nécessitant un GPU	Oui : nœuds Amazon Linux uniquement	Oui	Oui
Possibilité d'exécuter des applications nécessitant des processeurs Arm	Oui	Oui	Oui
Possibilité d'exécuter AWS Bottlerocket	Oui	Oui	Non
Les pods partagent le processeur, la mémoire, le stockage	Oui	Oui	Oui

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
et les ressources réseau avec d'autres pods.			
Doit déployer et gérer des EC2 instances Amazon	Oui	Non, en savoir plus sur les instances EC2 gérées	Oui, les machines physiques ou virtuelles sur site sont gérées par vous avec l'outillage de votre choix.
Doit sécuriser, maintenir et corriger le système d'exploitation des EC2 instances Amazon	Oui	Non	Oui, le système d'exploitation exécuté sur vos machines physiques ou virtuelles est géré par vous avec l'outillage de votre choix.
Peut fournir des arguments bootstrap lors du déploiement d'un nœud, tels que des arguments kubelet supplémentaires.	Oui — Utilisation `eksctl` ou lancement d'un modèle avec une AMI personnalisée.	Non, utilisez a `NodeClass` pour configurer les nœuds	Oui, vous pouvez personnaliser les arguments du bootstrap avec nodeadm. Consultez the section called "Nœuds hybrides nodeadm".

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Peut attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de l'adresse IP attribuée au nœud.	Oui : utilisation d'un modèle de lancement avec une AMI personnalisée. Pour de plus amples informations, veuillez consulter the section called "Modèles de lancement".	Non	Oui, tu voisthe section called "Configurer CNI".
Possibilité d'accéder au nœud via SSH	Oui	Non, découvrez comment résoudre les problèmes liés aux nœuds	Oui
Possibilité de déployer votre propre AMI personnalisée sur les nœuds	Oui : utilisation d'un modèle de lancement	Non	Oui
Possibilité de déployer votre propre CNI personnalisée sur les nœuds	Oui : utilisation d'un modèle de lancement avec une AMI personnalisée	Non	Oui

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Vous devez mettre à jour l'AMI du nœud par vous-même	Oui — Si vous avez déployé une AMI optimisée pour Amazon EKS, vous êtes averti dans la console Amazon EKS lorsque des mises à jour sont disponibles. Vous pouvez effectuer la mise à jour en un clic dans la console. Si vous avez déployé une AMI personnalisée, vous n'êtes pas averti dans la console Amazon EKS lorsque des mises à jour sont disponibles. Vous devez effectuer la mise à jour par vous-même.	Non	Oui, le système d'exploitation exécuté sur vos machines physiques ou virtuelles est géré par vous avec l'outillage de votre choix. Consultez the section called "Préparer le système d'exploitation".

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Vous devez mettre à jour la version du nœud Kubernetes par vous-même	Oui — Si vous avez déployé une AMI optimisée pour Amazon EKS, vous êtes averti dans la console Amazon EKS lorsque des mises à jour sont disponibles. Vous pouvez effectuer la mise à jour en un clic dans la console. Si vous avez déployé une AMI personnal isée, vous n'êtes pas averti dans la console Amazon EKS lorsque des mises à jour sont disponibles. Vous devez effectuer la mise à jour par vous-même.	Non	Oui, vous gérez les mises à niveau des nœuds hybrides avec l'outillage de votre choix ou avec `nodeadm`. Consultez the section called "Mettre à niveau les nœuds hybrides".
Peut utiliser le stockage Amazon EBS avec des pods	Oui	Oui, en tant que fonctionnalité intégrée. Découvrez comment créer une classe de stockage.	Non
Peut utiliser le stockage Amazon EFS avec des pods	Oui	Oui	Non

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Peut utiliser Amazon FSx pour le stockage de Lustre avec des pods	Oui	Oui	Non
Possibilité d'utiliser le Network Load Balancer pour les services	Oui	Oui	Oui, le type de cible doit être utilisé `ip`.
Les pods peuvent s'exécuter dans un sous-réseau public	Oui	Oui	Non, les modules s'exécutent dans un environnement sur site.
Possibilité d'attribuer différents groupes de sécurité VPC à des pods individuels	Oui : nœuds Linux uniquement	Non	Non
Peut exécuter Kubernetes DaemonSets	Oui	Oui	Oui
Support `HostPort` et manifeste `HostNetwork` intégré au Pod	Oui	Oui	Oui

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
AWS Disponibilité de la région	Toutes les régions prises en charge par Amazon EKS	Toutes les régions prises en charge par Amazon EKS	Toutes les régions prises en charge par Amazon EKS, à l'exception des régions AWS GovCloud (États-Unis) et de la Chine.
Peut exécuter des conteneurs sur des hôtes EC2 dédiés Amazon	Oui	Non	Non
Tarification	Coût d'une EC2 instance Amazon qui exécute plusieurs pods. Pour plus d'informations, consultez les EC2 tarifs Amazon.	Lorsque le mode automatique EKS est activé dans votre cluster, vous payez des frais distincts, en plus des frais d' EC2 instance standard, pour les instances lancées à l'aide de la capacité de calcul du mode automatique. Le montant varie en fonction du type d'instance lancé et de la AWS région dans laquelle se trouve votre cluster. Pour plus d'informations, consultez la tarification d'Amazon EKS.	Coût des nœuds hybrides (vCPU) par heure. Pour plus d'informations, consultez la tarification d'Amazon EKS.

Simplifiez le cycle de vie des nœuds avec des groupes de nœuds gérés

Les groupes de nœuds gérés par Amazon EKS automatisent le provisionnement et la gestion du cycle de vie des nœuds (EC2 instances Amazon) pour les clusters Amazon EKS Kubernetes.

Avec les groupes de nœuds gérés par Amazon EKS, vous n'avez pas besoin de configurer ou d'enregistrer séparément les EC2 instances Amazon qui fournissent la capacité de calcul pour exécuter vos applications Kubernetes. Vous pouvez créer, mettre à jour ou résilier les nœuds pour votre cluster en une seule opération. Les mises à jour et les résiliations de nœuds purgent automatiquement les nœuds afin de garantir la disponibilité de vos applications.

Chaque nœud géré est approvisionné dans le cadre d'un groupe Amazon EC2 Auto Scaling géré pour vous par Amazon EKS. Toutes les ressources, y compris les instances et les groupes Auto Scaling, s'exécutent au sein de votre AWS compte. Chaque groupe de nœuds s'exécute dans plusieurs zones de disponibilité que vous définissez.

Les groupes de nœuds gérés peuvent également éventuellement tirer parti de la réparation automatique des nœuds, qui surveille en permanence l'état de santé des nœuds. Il réagit automatiquement aux problèmes détectés et remplace les nœuds lorsque cela est possible. Cela permet une disponibilité globale du cluster avec une intervention manuelle minimale. Pour de plus amples informations, veuillez consulter the section called “Santé des nœuds”.

Vous pouvez ajouter un groupe de nœuds gérés à des clusters nouveaux ou existants à l'aide de la console, de la AWS CLI`eksctl`, de l' AWS API ou de l'infrastructure Amazon EKS en tant qu'outils de code, notamment AWS CloudFormation. Les nœuds lancés dans le cadre d'un groupe de nœuds géré sont automatiquement étiquetés pour être découverts automatiquement par le Kubernetes Cluster Autoscaler. Vous pouvez utiliser le groupe de nœuds pour appliquer des labels Kubernetes aux nœuds et les mettre à jour à tout moment.

L'utilisation des groupes de nœuds gérés par Amazon EKS n'entraîne aucun coût supplémentaire. Vous ne payez que pour les AWS ressources que vous fournissez. Il s'agit notamment EC2 des instances Amazon, des volumes Amazon EBS, des heures de cluster Amazon EKS et de toute autre AWS infrastructure. Aucun frais minimum ni aucun engagement initial ne s'appliquent.

Pour démarrer avec un nouveau cluster Amazon EKS et un groupe de nœuds gérés, consultez the section called “Création d'un cluster (console et CLI)”.

Pour ajouter un groupe de nœuds gérés à un cluster existant, consultez the section called “Création”.

Concepts des groupes de nœuds gérés

- Les groupes de nœuds gérés par Amazon EKS créent et gèrent EC2 des instances Amazon pour vous.
- Chaque nœud géré est approvisionné dans le cadre d'un groupe Amazon EC2 Auto Scaling géré pour vous par Amazon EKS. De plus, toutes les ressources, y compris les EC2 instances Amazon et les groupes Auto Scaling, s'exécutent au sein de votre AWS compte.
- Le groupe Auto Scaling d'un groupe de nœuds gérés couvre chaque sous-réseau que vous spécifiez lorsque vous créez le groupe.
- Amazon EKS balise les ressources des groupes de nœuds gérés afin qu'elles soient configurées pour utiliser le Kubernetes Cluster Autoscaler.

> **Important**
>
> Si vous exécutez une application dynamique dans plusieurs zones de disponibilité qui est soutenue par des volumes Amazon EBS et que vous utilisez le Kubernetes Cluster Autoscaler, vous devez configurer plusieurs groupes de nœuds, chacun limité à une seule zone de disponibilité. En outre, vous devez activer la fonction `--balance-similar-node-groups`.

- Vous pouvez utiliser un modèle de lancement personnalisé pour un plus grand niveau de flexibilité et de personnalisation lors du déploiement de nœuds gérés. Par exemple, vous pouvez spécifier des arguments `kubelet` supplémentaires et utiliser une AMI personnalisée. Pour de plus amples informations, veuillez consulter the section called "Modèles de lancement". Si vous n'utilisez pas de modèle de lancement personnalisé lors de la première création d'un groupe de nœuds gérés, il existe un modèle de lancement généré automatiquement. Ne modifiez pas manuellement ce modèle généré automatiquement, sinon des erreurs se produiront.
- Amazon EKS suit le modèle de responsabilité partagée CVEs et les correctifs de sécurité sur les groupes de nœuds gérés. Lorsque les nœuds gérés exécutent une AMI optimisée pour Amazon EKS, cette dernière est chargée de créer des versions corrigées de l'AMI lorsque des bogues ou des problèmes sont signalés. Nous pouvons publier un correctif. Cependant, vous êtes responsable du déploiement de ces versions d'AMI corrigées sur vos groupes de nœuds gérés. Lorsque des nœuds gérés exécutent une AMI personnalisée, vous êtes chargé de créer des versions corrigées de l'AMI lorsque des bogues ou des problèmes sont signalés, puis de déployer l'AMI. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour".

- Les groupes de nœuds gérés par Amazon EKS peuvent être lancés dans des sous-réseaux publics et privés. Si vous lancez un groupe de nœuds gérés dans un sous-réseau public à partir du 22 avril 2020, `MapPublicIpOnLaunch` du sous-réseau doit avoir la valeur true pour que les instances puissent rejoindre un cluster. Si le sous-réseau public a été créé à l'aide `eksctl` des AWS CloudFormation modèles Amazon EKS vendus le 26 mars 2020 ou après cette date, ce paramètre est déjà défini sur true. Si les sous-réseaux publics ont été créés avant le 26 mars 2020, vous devez modifier le paramètre manuellement. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau.
- Lorsque vous déployez un groupe de nœuds gérés dans des sous-réseaux privés, vous devez vous assurer qu'il peut accéder à Amazon ECR pour extraire des images de conteneurs. Vous pouvez procéder en connectant une passerelle NAT à la table de routage du sous-réseau ou en ajoutant les points de terminaison d'un VPC AWS PrivateLink suivants :
  - Interface de point de terminaison de l'API Amazon ECR : `com.amazonaws.`*`region-code`*`.ecr.api`
  - Interface de point de terminaison de l'API de registre Docker Amazon ECR : `com.amazonaws.`*`region-code`*`.ecr.dkr`
  - Point de terminaison de la passerelle Amazon S3 : `com.amazonaws.`*`region-code`*`.s3`

  Pour d'autres services et points de terminaison couramment utilisés, veuillez consulter la rubrique the section called "Clusters privés".
- Les groupes de nœuds gérés ne peuvent pas être déployés sur AWS Outposts ou dans Wavelength AWS . Les groupes de nœuds gérés peuvent être créés dans les Zones AWS Locales. Pour de plus amples informations, veuillez consulter the section called " AWS Zones Locales".
- Vous pouvez créer plusieurs groupes de nœuds gérés au sein d'un même cluster. Par exemple, vous pouvez créer un groupe de nœuds avec l'AMI Amazon Linux standard optimisée pour Amazon EKS pour certaines charges de travail et un autre avec la variante GPU pour les charges de travail nécessitant un support GPU.
- Si votre groupe de nœuds gérés rencontre un échec de vérification de l'état d'une EC2 instance Amazon, Amazon EKS renvoie un code d'erreur pour vous aider à diagnostiquer le problème. Pour de plus amples informations, veuillez consulter the section called "Codes d'erreurs liées aux groupes de nœuds gérés".
- Amazon EKS ajoute des labels Kubernetes aux instances de groupes de nœuds gérés. Ces labels fournis par Amazon EKS ont le préfixe `eks.amazonaws.com`.
- Amazon EKS purge automatiquement les nœuds à l'aide de l'API Kubernetes lors des résiliations ou des mises à jour.

- Les budgets d'interruption des pods ne sont pas respectés lorsqu'il s'agit de mettre fin à un nœud `AZRebalance` ou de réduire le nombre de nœuds souhaité. Ces actions tentent d'expulser les pods du nœud. Mais si cela prend plus de 15 minutes, le nœud est arrêté, que tous les pods du nœud soient interrompus ou non. Pour prolonger le délai d'arrêt du nœud, ajoutez un hook de cycle de vie au groupe Auto Scaling. Pour plus d'informations, consultez la section Ajouter des hooks de cycle de vie dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
- Pour exécuter correctement le processus de vidange après avoir reçu une notification d'interruption ponctuelle ou une notification de rééquilibrage des capacités, `CapacityRebalance` doit être réglé sur `true`.
- La mise à jour des groupes de nœuds gérés respecte les budgets d'interruption des pods que vous avez définis pour vos pods. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour les détails du comportement".
- L'utilisation de groupes de nœuds gérés par Amazon EKS est disponible sans coûts supplémentaires. Vous ne payez que pour les AWS ressources que vous fournissez.
- Si vous souhaitez chiffrer les volumes Amazon EBS pour vos nœuds, vous pouvez déployer les nœuds à l'aide d'un modèle de lancement. Pour déployer des nœuds gérés avec des volumes Amazon EBS chiffrés sans utiliser de modèle de lancement, chiffrez tous les nouveaux volumes Amazon EBS créés dans votre compte. Pour plus d'informations, consultez la section Chiffrement par défaut dans le guide de EC2 l'utilisateur Amazon.

Types de capacité des groupes de nœuds gérés

Lorsque vous créez un groupe de nœuds gérés, vous pouvez choisir le type de capacité à la demande ou Spot. Amazon EKS déploie un groupe de nœuds gérés avec un groupe Amazon EC2 Auto Scaling qui contient uniquement des instances On-Demand ou uniquement des instances Amazon EC2 Spot. Vous pouvez planifier des pods pour les applications tolérantes aux pannes vers des groupes de nœuds gérés par Spot, et des applications intolérantes aux pannes vers des groupes de nœuds à la demande au sein d'un seul cluster Kubernetes. Par défaut, un groupe de nœuds gérés déploie les EC2 instances Amazon On-Demand.

À la demande

Avec les instances à la demande, vous payez la capacité de calcul à la seconde, sans engagement à long terme.

Par défaut, si vous ne spécifiez pas de Type de capacité, le groupe de nœuds gérés est alloué avec des instances à la demande. Un groupe de nœuds géré configure un groupe Amazon EC2 Auto Scaling en votre nom avec les paramètres suivants appliqués :

- La stratégie d'allocation pour fournir la capacité à la demande est définie sur `prioritized`. Les groupes de nœuds gérés utilisent l'ordre des types d'instance transmis dans l'API pour déterminer le type d'instance à utiliser en premier lors de la fourniture de la capacité à la demande. Par exemple, vous pouvez spécifier trois types d'instances dans l'ordre suivant : `c5.large`, `c4.large` et `c3.large`. Lorsque vos instances à la demande sont lancées, le groupe de nœuds gérés fournit la capacité à la demande en commençant par `c5.large`, puis `c4.large` et enfin `c3.large`. Pour plus d'informations, consultez le groupe Amazon EC2 Auto Scaling dans le guide de l'utilisateur Amazon EC2 Auto Scaling.
- Amazon EKS ajoute le label Kubernetes suivant à tous les nœuds de votre groupe de nœuds gérés qui spécifie le type de capacité : `eks.amazonaws.com/capacityType: ON_DEMAND` Vous pouvez utiliser ce label pour programmer des applications à état ou intolérantes aux pannes sur des nœuds à la demande.

Spot

Les instances Amazon EC2 Spot sont des EC2 capacités inutilisées d'Amazon qui offrent des remises importantes par rapport aux prix à la demande. Les instances Amazon EC2 Spot peuvent être interrompues moyennant un préavis de deux minutes lorsqu'il est EC2 nécessaire de rétablir leur capacité. Pour plus d'informations, consultez la section Instances Spot dans le guide de EC2 l'utilisateur Amazon. Vous pouvez configurer un groupe de nœuds gérés avec des instances Amazon EC2 Spot afin d'optimiser les coûts des nœuds de calcul exécutés dans votre cluster Amazon EKS.

Pour utiliser des instances Spot dans un groupe de nœuds gérés, créez un groupe de nœuds gérés en définissant le type de capacité comme `spot`. Un groupe de nœuds géré configure un groupe Amazon EC2 Auto Scaling en votre nom en appliquant les meilleures pratiques Spot suivantes :

- Pour garantir que vos nœuds Spot sont alloués dans les groupes de capacité Spot optimaux, la stratégie d'allocation est définie sur l'une des options suivantes :
  - `price-capacity-optimized`(PCO) — Lors de la création de nouveaux groupes de nœuds dans un cluster doté d'une version de Kubernetes `1.28` ou supérieure, la stratégie d'allocation est définie sur. `price-capacity-optimized` Toutefois, la stratégie d'allocation ne sera pas modifiée pour les groupes de nœuds déjà créés avec `capacity-optimized` avant que les groupes de nœuds gérés par Amazon EKS ne commencent à prendre en charge le PCO.

- `capacity-optimized`(CO) — Lors de la création de nouveaux groupes de nœuds dans un cluster avec une version de Kubernetes `1.27` ou inférieure, la stratégie d'allocation est définie sur. `capacity-optimized`

  Pour augmenter le nombre de groupes de capacité Spot disponibles pour l'allocation de capacité, configurez un groupe de nœuds gérés pour utiliser plusieurs types d'instances.

- Le rééquilibrage de capacité Amazon EC2 Spot est activé afin qu'Amazon EKS puisse drainer et rééquilibrer en douceur vos nœuds Spot afin de minimiser les perturbations des applications lorsqu'un nœud Spot présente un risque élevé d'interruption. Pour plus d'informations, consultez Amazon EC2 Auto Scaling Capacity Rebalancing dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
  - Lorsqu'un nœud Spot reçoit une recommandation de rééquilibrage, Amazon EKS tente automatiquement de lancer un nouveau nœud Spot de remplacement.
  - Si un avis d'interruption de deux minutes arrive avant que le nœud Spot de remplacement ne soit dans l'état `Ready`, Amazon EKS commence à purger le nœud Spot qui a reçu la recommandation de rééquilibrage. Amazon EKS vide le nœud dans la mesure du possible. Par conséquent, rien ne garantit qu'Amazon EKS attendra que le nœud de remplacement rejoigne le cluster avant de vider le nœud existant.
  - Lorsqu'un nœud Spot de remplacement est démarré et est dans l'état `Ready` sur Kubernetes, Amazon EKS cloisonne et purge le nœud Spot qui a reçu la recommandation de rééquilibrage. Le fait de boucler le nœud Spot garantit que le contrôleur de service n'envoie aucune nouvelle demande à ce nœud Spot. Il le supprime également de sa liste de nœuds Spot sains et actifs. Le drainage du nœud Spot garantit que les pods en cours d'exécution sont expulsés en toute élégance.
- Amazon EKS ajoute le label Kubernetes suivant à tous les nœuds de votre groupe de nœuds gérés qui spécifie le type de capacité : `eks.amazonaws.com/capacityType: SPOT` Vous pouvez utiliser ce label pour programmer des applications tolérantes aux pannes sur les nœuds Spot.

Lorsque vous décidez de déployer un groupe de nœuds avec une capacité à la demande ou Spot, vous devez tenir compte des conditions suivantes :

- Les instances Spot conviennent bien aux applications sans état, tolérantes aux pannes et flexibles. Il s'agit notamment des charges de travail de formation par lots et de machine learning, des mégadonnées ETLs telles qu'Apache Spark, des applications de traitement des files d'attente et des points de terminaison d'API sans état. Le Spot étant une EC2 capacité inutilisée d'Amazon, qui peut changer au fil du temps, nous vous recommandons d'utiliser la capacité Spot pour les charges

de travail tolérantes aux interruptions. Plus précisément, la capacité Spot convient aux charges de travail qui peuvent tolérer des périodes pendant lesquelles la capacité requise n'est pas disponible.

- Nous vous recommandons d'utiliser la capacité à la demande pour les applications qui sont intolérantes aux pannes. Cela inclut les outils de gestion de cluster tels que les outils de surveillance et d'exploitation, les déploiements qui nécessitent `StatefulSets`, et les applications avec état, telles que les bases de données.
- Pour maximiser la disponibilité de vos applications en utilisant les instances Spot, nous vous recommandons de configurer un groupe de nœuds gérés Spot pour utiliser plusieurs types d'instances. Nous vous recommandons d'appliquer les règles suivantes lorsque vous utilisez plusieurs types d'instance :
  - Au sein d'un groupe de nœuds gérés, si vous utilisez le Cluster Autoscaler, nous vous recommandons d'utiliser un ensemble flexible de types d'instances avec la même quantité de vCPU et de ressources de mémoire. Ceci afin de garantir que les nœuds de votre cluster soient mis à l'échelle comme prévu. Par exemple, si vous avez besoin de quatre V CPUs et de huit GiB de mémoire, utilisez`c3.xlarge`,,`c4.xlarge`,`c5.xlarge`, `c5d.xlarge` `c5a.xlarge`c`5n.xlarge`, ou d'autres types d'instance similaires.
  - Pour améliorer la disponibilité des applications, nous recommandons de déployer plusieurs groupes de nœuds gérés Spot. Pour cela, chaque groupe doit utiliser un ensemble flexible de types d'instances qui disposent des mêmes ressources vCPU et mémoire. Par exemple, si vous avez besoin de 4 V CPUs et 8 GiB de mémoire, nous vous recommandons de créer un groupe de nœuds gérés avec`c3.xlarge`,,`c4.xlarge`,`c5.xlarge`, `c5d.xlarge` `c5a.xlarge`c`5n.xlarge`, ou d'autres types d'instances similaires, et un second groupe de nœuds gérés avec`m3.xlarge`,,, `m4.xlarge` `m5.xlarge` `m5d.xlarge`m`5a.xlarge`, `m5n.xlarge` ou d'autres types d'instances similaires.
  - Lorsque vous déployez votre groupe de nœuds avec le type de capacité Spot qui utilise un modèle de lancement personnalisé, utilisez l'API pour transmettre plusieurs types d'instances. Ne transmettez pas un seul type d'instance dans le modèle de lancement. Pour plus d'informations sur le déploiement d'un groupe de nœuds à l'aide d'un modèle de lancement, consultez the section called "Modèles de lancement".

Créez un groupe de nœuds gérés pour votre cluster

Cette rubrique explique comment lancer des groupes de nœuds gérés Amazon EKS composés de nœuds qui s'enregistrent dans votre cluster Amazon EKS. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes.

Si c'est la première fois que vous lancez un groupe de nœuds géré par Amazon EKS, nous vous recommandons de suivre l'un de nos guidesMise en route. Ces guides fournissent des instructions détaillées pour créer un cluster Amazon EKS avec des nœuds.

> **Important**
>
> - Les nœuds Amazon EKS sont des EC2 instances Amazon standard. Vous êtes facturé sur la base des EC2 prix habituels d'Amazon. Pour plus d'informations, consultez Amazon EC2 Pricing.
> - Vous ne pouvez pas créer de nœuds gérés dans une AWS région où AWS Outposts ou AWS Wavelength sont activés. Vous pouvez créer des nœuds autogérés à la place. Pour plus d'informations, consultez the section called "Amazon Linux", the section called "Windows" et the section called "Bottlerocket". Vous pouvez également créer un groupe de nœuds Amazon Linux autogéré sur un Outpost. Pour de plus amples informations, veuillez consulter the section called "Nœuds".
> - Si vous ne spécifiez pas d'ID AMI pour le `bootstrap.sh` fichier inclus dans Linux ou Bottlerocket optimisés pour Amazon EKS, les groupes de nœuds gérés imposent un nombre maximum sur la valeur de. `maxPods` Pour les instances de moins de 30 VCPUs, le nombre maximum est de`110`. Pour les instances avec une tension supérieure à 30 VCPUs, le nombre maximum passe à`250`. Ces chiffres sont basés sur les seuils d'évolutivité de Kubernetes et les paramètres recommandés par l'équipe interne d'Amazon EKS chargée des tests d'évolutivité. Pour plus d'informations, consultez l'article de blog Plugin CNI Amazon VPC augmente les limites du nombre de pods par nœud (français non garanti).

- Un cluster Amazon EKS existant. Pour en déployer un, consultez the section called "Créer un cluster ".
- Un rôle IAM existant pour les nœuds à utiliser. Pour en créer un, consultez the section called "Rôle IAM de nœud". Si ce rôle ne possède aucune des politiques du VPC CNI, le rôle distinct qui suit est requis pour les pods CNI du VPC.
- (Facultatif, mais recommandé) Le module complémentaire Amazon VPC CNI pour Kubernetes est configuré avec son propre rôle IAM auquel est attachée la politique IAM nécessaire. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
- Connaissance des considérations répertoriées dans la section Choisissez un type d'instance de EC2 nœud Amazon optimal. Selon le type d'instance que vous choisissez, il peut y avoir des prérequis supplémentaires pour votre cluster et votre VPC.

- Pour ajouter un groupe de nœuds gérés par Windows, vous devez d'abord activer le support Windows pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Activer le support Windows".

Vous pouvez créer un groupe de nœuds gérés avec l'une des options suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Créez un groupe de nœuds géré avec eksctl

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

1. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy est attachée au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
2. Créez un groupe de nœuds gérés avec ou sans modèle de lancement personnalisé. La spécification manuelle d'un modèle de lancement permet une plus grande personnalisation d'un groupe de nœuds. Par exemple, cela peut permettre de déployer une AMI personnalisée ou de fournir des arguments au script `boostrap.sh` dans une AMI optimisée par Amazon EKS. Pour obtenir la liste complète de toutes les options disponibles et des valeurs par défaut, saisissez la commande suivante.

   ```
   eksctl create nodegroup --help
   ```

   Dans la commande suivante, remplacez *`my-cluster`* par le nom de votre cluster et remplacez *`my-mng`* par le nom de votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas

comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

> **Important**
>
> Si vous n'utilisez pas de modèle de lancement personnalisé lors de la première création d'un groupe de nœuds gérés, n'en utilisez pas un ultérieurement pour le groupe de nœuds. Si vous n'avez pas spécifié de modèle de lancement personnalisé, le système génère automatiquement un modèle de lancement qu'il est déconseillé de modifier manuellement. La modification manuelle de ce modèle de lancement généré automatiquement peut entraîner des erreurs.

Sans modèle de lancement

`eksctl`crée un modèle de EC2 lancement Amazon par défaut dans votre compte et déploie le groupe de nœuds à l'aide d'un modèle de lancement créé en fonction des options que vous spécifiez. Avant de spécifier une valeur pour `--node-type`, consultez the section called "Types d' EC2 instances Amazon".

Remplacez *`ami-family`* par un mot clé autorisé. Pour plus d'informations, consultez Définition de la famille AMI de nœuds dans la documentation `eksctl`. *`my-key`*Remplacez-le par le nom de votre paire de EC2 clés Amazon ou de votre clé publique. Cette clé est utilisée pour SSH dans vos nœuds après leur lancement.

> **Note**
>
> Pour Windows, cette commande n'active pas SSH. Au lieu de cela, il associe votre paire de EC2 clés Amazon à l'instance et vous permet d'accéder à l'instance par RDP.

Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations sur Linux, consultez les paires de EC2 clés Amazon et les instances Linux dans le guide de EC2 l'utilisateur Amazon. Pour plus d'informations sur Windows, consultez les paires de EC2 clés Amazon et les instances Windows dans le guide de EC2 l'utilisateur Amazon.

Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :

- Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
- Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vous souhaitez bloquer l'accès du Pod à l'IMDS, ajoutez l'`--disable-pod-imds`option à la commande suivante.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key my-key
```

Vos instances peuvent éventuellement attribuer un nombre nettement plus élevé d'adresses IP aux pods, attribuer des adresses IP aux pods provenant d'un bloc CIDR différent de celui de l'instance et être déployées sur un cluster sans accès à Internet. Pour plus d'informations, consultez the section called "Augmenter le nombre d'adresses IP", the section called "Mise en réseau personnalisée" et the section called "Clusters privés" pour connaître les options supplémentaires à ajouter à la commande précédente.

Les groupes de nœuds gérés calculent et appliquent une valeur unique pour le nombre maximum de pods pouvant être exécutés sur chaque nœud de votre groupe de nœuds, en fonction du type d'instance. Si vous créez un groupe de nœuds avec différents types d'instances, la plus petite valeur calculée pour tous les types d'instances est appliquée en tant que nombre maximum de pods pouvant être exécutés sur chaque type d'instance du groupe de nœuds. Les groupes de nœuds gérés calculent la valeur à l'aide du script référencé dans le nombre maximal de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon.

Avec un modèle de lancement

Le modèle de lancement doit déjà exister et répondre aux exigences spécifiées dans Notions de base de configuration du modèle de lancement. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :

- Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
- Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vous souhaitez bloquer l'accès du Pod à l'IMDS, spécifiez les paramètres nécessaires dans le modèle de lancement.

a. Copiez les contenus suivants sur votre appareil. Remplacez les *example values* puis exécutez la commande modifiée pour créer le fichier `eks-nodegroup.yaml`. Plusieurs paramètres que vous spécifiez lors d'un déploiement sans modèle de lancement sont déplacés dans le modèle de lancement. Si vous ne spécifiez pas de`version`, la version par défaut du modèle est utilisée.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

Pour obtenir la liste complète des paramètres du fichier de configuration `eksctl`, consultez Schéma de fichier de configuration dans la documentation `eksctl`. Vos instances peuvent éventuellement attribuer un nombre nettement plus élevé d'adresses IP aux pods, attribuer des adresses IP aux pods provenant d'un bloc CIDR différent de celui de l'instance et être déployées sur un cluster sans accès Internet sortant. Pour plus d'informations, consultez the section called "Augmenter le nombre d'adresses IP"the section called "Mise en réseau personnalisée", et

the section called "Clusters privés" pour les options supplémentaires à ajouter au fichier de configuration.

Si vous n'avez pas spécifié d'ID AMI dans votre modèle de lancement, les groupes de nœuds gérés calculent et appliquent une valeur unique pour le nombre maximum de pods pouvant être exécutés sur chaque nœud de votre groupe de nœuds, en fonction du type d'instance. Si vous créez un groupe de nœuds avec différents types d'instances, la plus petite valeur calculée pour tous les types d'instances est appliquée en tant que nombre maximum de pods pouvant être exécutés sur chaque type d'instance du groupe de nœuds. Les groupes de nœuds gérés calculent la valeur à l'aide du script référencé dans le nombre maximal de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon.

Si vous avez spécifié un ID AMI dans votre modèle de lancement, spécifiez le nombre maximum de pods pouvant être exécutés sur chaque nœud de votre groupe de nœuds si vous utilisez un réseau personnalisé ou si vous souhaitez augmenter le nombre d'adresses IP attribuées à votre instance. Pour de plus amples informations, veuillez consulter the section called "Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon".

b. Déployez le groupe de nœuds avec la commande suivante.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

Créez un groupe de nœuds géré à l'aide du AWS Management Console

1. Attendez que le statut de votre cluster s'affiche soit `ACTIVE`. Vous ne pouvez pas créer de groupe de nœuds gérés pour un cluster qui ne l'est pas déjà`ACTIVE`.
2. Ouvrez la console Amazon EKS.
3. Choisissez le nom du cluster dans lequel vous souhaitez créer un groupe de nœuds gérés.
4. Sélectionner l'onglet Calcul.
5. Choisissez Ajouter un groupe de nœuds.
6. Sur la page Configurer un groupe de nœuds, définissez les paramètres en conséquence, puis choisissez Next (Suivant).
   - Nom : saisissez un nom unique pour votre groupe de nœuds gérés. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un

chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

- Rôle IAM de nœud : choisissez le rôle d'instance de nœud à utiliser avec votre groupe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM de nœud".

> **Important**
>
> - Vous ne pouvez pas utiliser le même rôle que celui utilisé pour créer des clusters.
> - Nous vous recommandons d'utiliser un rôle qui n'est actuellement utilisé par aucun groupe de nœuds autogéré. Sinon, vous prévoyez de l'utiliser avec un nouveau groupe de nœuds autogérés. Pour de plus amples informations, veuillez consulter the section called "Suppression".

- Utiliser le modèle de lancement : (facultatif) choisissez si vous souhaitez utiliser un modèle de lancement existant. Sélectionnez un Nom de modèle de lancement. Sélectionnez ensuite une Version du modèle de lancement. Si vous ne sélectionnez aucune version, Amazon EKS utilise la version par défaut du modèle. Les modèles de lancement permettent de personnaliser davantage votre groupe de nœuds, par exemple en vous permettant de déployer une AMI personnalisée, d'attribuer un nombre nettement plus élevé d'adresses IP aux pods, d'attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de celui de l'instance et de déployer des nœuds sur un cluster sans accès Internet sortant. Pour plus d'informations, consultez the section called "Augmenter le nombre d'adresses IP", the section called "Mise en réseau personnalisée" et the section called "Clusters privés".

  Le modèle de lancement doit répondre aux exigences de la section Personnaliser les nœuds gérés avec des modèles de lancement. Si vous n'utilisez pas votre propre modèle de lancement, l'API Amazon EKS crée un modèle de EC2 lancement Amazon par défaut dans votre compte et déploie le groupe de nœuds à l'aide du modèle de lancement par défaut.

  Si vous implémentez des rôles IAM pour les comptes de service, si vous attribuez les autorisations nécessaires directement à chaque pod nécessitant un accès aux AWS services, et si aucun pod de votre cluster n'a besoin d'accéder à IMDS pour d'autres raisons, telles que la récupération de la AWS région actuelle, vous pouvez également désactiver l'accès à IMDS pour les pods qui n'utilisent pas le réseau hôte dans un modèle de lancement. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

- Labels Kubernetes : (facultatif) vous pouvez choisir d'appliquer des labels Kubernetes aux nœuds de votre groupe de nœuds gérés.
- Nuances Kubernetes — (Facultatif) Vous pouvez choisir d'appliquer des taches Kubernetes aux nœuds de votre groupe de nœuds géré. Les options disponibles dans le menu Effet sont **`NoSchedule`** , **`NoExecute`** , et **`PreferNoSchedule`** . Pour de plus amples informations, veuillez consulter the section called "Empêcher la planification des pods sur des nœuds spécifiques".
- Identifications : (facultatif) vous pouvez choisir d'identifier votre groupe de nœuds gérés Amazon EKS. Ces balises ne se propagent pas aux autres ressources du groupe de nœuds, telles que les groupes ou les instances Auto Scaling. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ".

7. Sur la page Définir la configuration de calcul et de mise à l'échelle, définissez les paramètres en conséquence, puis choisissez Next (Suivant).
   - Type d'AMI — Sélectionnez un type d'AMI. Si vous déployez des instances Arm, n'oubliez pas de prendre en compte les points à prendre en compte dans Arm Amazon Linux optimisé pour Amazon EKS AMIs avant le déploiement.

     Si vous avez spécifié un modèle de lancement sur la page précédente et que vous avez indiqué une AMI dans le modèle de lancement, vous ne pouvez pas sélectionner de valeur. La valeur du modèle s'affiche. L'AMI spécifiée dans le modèle doit répondre aux exigences de la section Spécification d'une AMI.
   - Type de capacité : sélectionnez un type de capacité. Pour plus d'informations sur le choix d'un type de capacité, consultez the section called "Types de capacité des groupes de nœuds gérés". Vous ne pouvez pas mélanger différents types de capacité au sein d'un même groupe de nœuds. Si vous souhaitez utiliser les deux types de capacités, créez des groupes de nœuds distincts, chacun avec son propre type de capacité et d'instance. Consultez la section Réserver GPUs pour les groupes de nœuds gérés pour obtenir des informations sur le provisionnement et le dimensionnement des nœuds de travail accélérés par GPU.
   - Types d'instance : par défaut, un ou plusieurs types d'instances sont spécifiés. Pour supprimer un type d'instance par défaut, sélectionnez la croix (X) sur le côté droit du type d'instance. Choisissez les types d'instances à utiliser dans votre groupe de nœuds gérés. Pour de plus amples informations, veuillez consulter the section called "Types d' EC2 instances Amazon".

     La console affiche un ensemble de types d'instances couramment utilisés. Si vous devez créer un groupe de nœuds gérés avec un type d'instance qui n'est pas affiché`eksctl`, utilisez la AWS CLI ou un SDK pour créer le groupe de nœuds. AWS CloudFormation Si vous avez spécifié

un modèle de lancement sur la page précédente, vous ne pouvez pas sélectionner de valeur car le type d'instance doit être spécifié dans le modèle de lancement. La valeur du modèle de lancement s'affiche. Si vous avez sélectionné Spot pour le type de capacité, nous vous recommandons de spécifier plusieurs types d'instances pour améliorer la disponibilité.

- Taille du disque — Entrez la taille du disque (en GiB) à utiliser pour le volume racine de votre nœud.

  Si vous avez spécifié un modèle de lancement sur la page précédente, vous ne pouvez pas sélectionner de valeur car elle doit être spécifiée dans le modèle de lancement.

- Taille souhaitée : spécifiez le nombre actuel de nœuds que le groupe de nœuds gérés doit conserver au lancement.

  > **Note**
  >
  > Amazon EKS ne redimensionne pas automatiquement votre groupe de nœuds. Cependant, vous pouvez configurer le Kubernetes Cluster Autoscaler pour le faire à votre place. Pour plus d'informations, consultez Cluster Autoscaler activé. AWS

- Taille minimale : spécifiez le nombre minimal de nœuds vers lequel le groupe de nœuds gérés peut être mis à l'échelle.
- Taille maximale : spécifiez le nombre maximal de nœuds vers lequel le groupe de nœuds gérés peut être mis à niveau.
- Configuration des mises à jour du groupe : (facultatif) vous pouvez sélectionner le nombre ou le pourcentage de nœuds à mettre à jour en parallèle. Ces nœuds seront indisponibles pendant la mise à jour. Pour Maximum non disponible, sélectionnez l'une des options suivantes et spécifiez une valeur :
  - Nombre : sélectionnez et spécifiez le nombre de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle.
  - Pourcentage : sélectionnez et spécifiez le pourcentage de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle. Cela est pratique si votre groupe de nœuds contient de nombreux nœuds.
- Configuration de la réparation automatique des nœuds — (Facultatif) Si vous cochez la case Activer la réparation automatique des nœuds, Amazon EKS remplacera automatiquement les nœuds lorsque des problèmes seront détectés. Pour de plus amples informations, veuillez consulter the section called “Santé des nœuds”.

8. Sur la page Spécifier les détails, définissez les paramètres en conséquence, puis choisissez Next (Suivant).
   - Sous-réseaux : choisissez les sous-réseaux dans lesquels vous souhaitez lancer vos nœuds gérés.

> **Important**
>
> Si vous exécutez une application dynamique dans plusieurs zones de disponibilité qui est soutenue par des volumes Amazon EBS et que vous utilisez le Kubernetes Cluster Autoscaler, vous devez configurer plusieurs groupes de nœuds, chacun limité à une seule zone de disponibilité. En outre, vous devez activer la fonction `--balance-similar-node-groups`.

> **Important**
>
> - Si vous choisissez un sous-réseau public et que seul le point de terminaison du serveur d'API publique est activé, le paramètre `MapPublicIPOnLaunch` du sous-réseau doit avoir la valeur `true` pour que les instances rejoignent un cluster. Si le sous-réseau a été créé à l'aide `eksctl` des AWS CloudFormation modèles vendus par Amazon EKS le 26 mars 2020 ou après cette date, ce paramètre est déjà défini sur. `true` Si les sous-réseaux ont été créés avec les AWS CloudFormation modèles `eksctl` ou avant le 26 mars 2020, vous devez modifier le paramètre manuellement. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau.
> - Si vous utilisez un modèle de lancement et que vous spécifiez plusieurs interfaces réseau, Amazon EC2 n'attribuera pas automatiquement d'`IPv4`adresse publique, même si elle `MapPublicIpOnLaunch` est définie sur. `true` Pour que les nœuds rejoignent le cluster dans ce scénario, vous devez soit activer le point de terminaison du serveur d'API privé du cluster, soit lancer des nœuds dans un sous-réseau privé avec un accès Internet sortant fourni via une autre méthode, telle qu'une passerelle NAT. Pour plus d'informations, consultez la section Adressage IP des EC2 instances Amazon dans le guide de EC2 l'utilisateur Amazon.

   - Configurez l'accès SSH aux nœuds (facultatif). L'activation de SSH vous permet de vous connecter à vos instances et de recueillir des informations de diagnostic en cas de problème.

Nous vous recommandons vivement d'activer l'accès à distance lorsque vous créez un groupe de nœuds. Vous ne pouvez pas activer l'accès à distance une fois le groupe de nœuds créé.

Si vous avez choisi d'utiliser un modèle de lancement, cette option n'est pas affichée. Pour autoriser l'accès à distance pour vos noeuds, spécifiez une paire de clés dans le modèle de lancement et assurez-vous que le port approprié est ouvert aux nœuds des groupes de sécurité que vous spécifiez dans le modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Utilisation des groupes de sécurité".

> **Note**
>
> Pour Windows, cette commande n'active pas SSH. Au lieu de cela, il associe votre paire de EC2 clés Amazon à l'instance et vous permet d'accéder à l'instance par RDP.

- Pour la paire de clés SSH (facultatif), choisissez une clé Amazon EC2 SSH à utiliser. Pour plus d'informations sur Linux, consultez les paires de EC2 clés Amazon et les instances Linux dans le guide de EC2 l'utilisateur Amazon. Pour plus d'informations sur Windows, consultez les paires de EC2 clés Amazon et les instances Windows dans le guide de EC2 l'utilisateur Amazon. Si vous avez choisi d'utiliser un modèle de lancement, vous ne pouvez pas en sélectionner un. Lorsqu'une clé Amazon EC2 SSH est fournie à des groupes de nœuds utilisant Bottlerocket AMIs, le conteneur administratif est également activé. Pour plus d'informations, consultez Conteneur d'administration sur GitHub.
- Pour Autoriser l'accès à distance SSH depuis, si vous souhaitez limiter l'accès à des instances spécifiques, sélectionnez les groupes de sécurité associés à ces instances. Si vous ne sélectionnez aucun groupe de sécurité spécifique, l'accès SSH est autorisé depuis n'importe où sur Internet (`0.0.0.0/0`).

9. Sur la page Review and create (Vérifier et créer), vérifiez la configuration de votre groupe de nœuds gérés et choisissez Create (Créer).

   Si les nœuds ne parviennent pas à rejoindre le cluster, reportez-vous the section called "Les nœuds ne parviennent pas à joindre le cluster" au chapitre Dépannage.

10. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

11. (Nœuds GPU uniquement) Si vous avez choisi un type d'instance GPU et une AMI accélérée optimisée pour Amazon EKS, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes

DaemonSet sur votre cluster. *vX.X.X*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Installer les modules complémentaires Kubernetes

Maintenant que vous disposez d'un cluster Amazon EKS fonctionnel avec des nœuds, vous êtes prêt à commencer à installer les modules complémentaires Kubernetes et à déployer des applications sur votre cluster. Les rubriques suivantes de la documentation vous aideront à étendre les fonctionnalités de votre cluster.

- Le principal IAM qui a créé le cluster est le seul principal capable de passer des appels au serveur d'API Kubernetes avec ou avec le. `kubectl` AWS Management Console Si vous souhaitez que d'autres principaux IAM aient accès à votre cluster, vous devez les ajouter. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Autorisations requises".
- Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :
  - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
  - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

  Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.
- Configurez le Kubernetes Cluster Autoscaler pour ajuster automatiquement le nombre de nœuds dans vos groupes de nœuds.
- Déployez un exemple d'application sur votre cluster.
- Organisez et surveillez les ressources du cluster à l'aide d'outils importants pour gérer votre cluster.

Mettre à jour un groupe de nœuds gérés pour votre cluster

Lorsque vous lancez une mise à jour d'un groupe de nœuds gérés, Amazon EKS met automatiquement à jour vos nœuds pour vous, en suivant les étapes répertoriées dans Comprendre

chaque phase des mises à jour des nœuds. Si vous utilisez une AMI optimisée pour Amazon EKS, Amazon EKS applique automatiquement les derniers correctifs de sécurité et mises à jour du système d'exploitation à vos nœuds dans le cadre de la dernière version de l'AMI.

Il existe plusieurs scénarios dans lesquels il est utile de mettre à jour la version ou la configuration de votre groupe de nœuds gérés Amazon EKS :

- Vous avez mis à jour la version de Kubernetes de votre cluster Amazon EKS et vous souhaitez mettre à jour vos nœuds pour utiliser la même version de Kubernetes.
- Une nouvelle version d'AMI est disponible pour votre groupe de nœuds gérés. Pour plus d'informations sur les versions d'AMI, consultez ces sections :
  - the section called "Obtenir des informations sur la version"
  - the section called "Bottlerocket"
  - the section called "Obtenir des informations sur la version"
- Vous souhaitez ajuster le nombre minimum, maximum ou souhaité d'instances dans votre groupe de nœuds gérés.
- Vous voulez ajouter ou supprimer les labels Kubernetes des instances de votre groupe de nœuds gérés.
- Vous souhaitez ajouter ou supprimer des AWS balises dans votre groupe de nœuds gérés.
- Vous devez déployer une nouvelle version d'un modèle de lancement avec des modifications de configuration, telles qu'une AMI personnalisée mise à jour.
- Vous avez déployé une version `1.9.0` ou une version ultérieure du module complémentaire Amazon VPC CNI, activé le module complémentaire pour la délégation de préfixes et souhaitez que les nouvelles instances de AWS Nitro System dans un groupe de nœuds prennent en charge un nombre considérablement accru de pods. Pour de plus amples informations, veuillez consulter the section called "Augmenter le nombre d'adresses IP".
- Vous avez activé la délégation de préfixes IP pour les nœuds Windows et vous souhaitez que les nouvelles instances AWS Nitro System d'un groupe de nœuds prennent en charge un nombre considérablement accru de pods. Pour de plus amples informations, veuillez consulter the section called "Augmenter le nombre d'adresses IP".

S'il existe une version plus récente de l'AMI pour la version Kubernetes de votre groupe de nœuds gérés, vous pouvez mettre à jour la version de votre groupe de nœuds pour utiliser la nouvelle version de l'AMI. De même, si votre cluster exécute une version de Kubernetes plus récente que

celle de votre groupe de nœuds, vous pouvez mettre à jour le groupe de nœuds pour qu'il utilise la dernière version de l'AMI afin qu'elle corresponde à la version Kubernetes de votre cluster.

Lorsqu'un nœud d'un groupe de nœuds géré est arrêté en raison d'une opération de dimensionnement ou d'une mise à jour, les pods de ce nœud sont d'abord vidés. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour les détails du comportement".

Mise à jour d'une version de groupe de nœuds

Vous pouvez mettre à jour la version d'un groupe de nœuds avec l'une des options suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

La version vers laquelle vous effectuez la mise à jour ne peut pas être supérieure à la version du plan de contrôle.

`eksctl`

Mettez à jour un groupe de nœuds gérés à l'aide de **`eksctl`**

Mettez à jour un groupe de nœuds gérés vers la dernière version de l'AMI de la même version de Kubernetes qui est actuellement déployée sur les nœuds à l'aide de la commande suivante. Remplacez chaque *`example value`* par vos propres valeurs.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> Si vous mettez à niveau un groupe de nœuds déployé avec un modèle de lancement vers une nouvelle version du modèle de lancement, ajoutez `--launch-template-version` *`version-number`* la commande précédente. Le modèle de lancement doit répondre aux exigences décrites dans Personnaliser les nœuds gérés avec des modèles de lancement. Si le modèle de lancement inclut une AMI personnalisée, celle-ci doit répondre aux exigences de la section Spécification d'une AMI. Lorsque vous mettez à niveau votre groupe de nœuds

vers une version plus récente de votre modèle de lancement, chaque nœud est recyclé pour correspondre à la nouvelle configuration de la version du modèle de lancement spécifiée.

Vous ne pouvez pas directement mettre à niveau un groupe de nœuds déployé sans modèle de lancement vers une nouvelle version du modèle de lancement. À la place, vous devez déployer un nouveau groupe de nœuds en utilisant le modèle de lancement pour mettre à jour le groupe de nœuds vers une nouvelle version du modèle de lancement.

Vous pouvez mettre à niveau un groupe de nœuds vers la même version que la version Kubernetes du plan de contrôle. Par exemple, si vous avez un cluster exécutant Kubernetes`1.33`, vous pouvez mettre à niveau les nœuds exécutant actuellement Kubernetes vers la version `1.32` à l'aide de la commande suivante. `1.33`

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.33
```

AWS Management Console

Mettez à jour un groupe de nœuds gérés à l'aide du AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster qui contient le groupe de nœuds à mettre à jour.
3. Si une mise à jour est disponible pour au moins un groupe de nœuds, une boîte de dialogue apparaît en haut de la page pour vous en informer. Si vous sélectionnez l'onglet Calculer, le message Mettre à jour maintenant s'affiche dans la colonne Version de version de l'AMI du tableau des groupes de nœuds pour lequel une mise à jour est disponible. Pour mettre à jour le groupe de nœuds, sélectionnez Update now (Mettre à jour maintenant).

   Vous ne verrez pas de notification pour les groupes de nœuds déployés avec une AMI personnalisée. Si vos nœuds sont déployés avec une AMI personnalisée, effectuez les étapes suivantes pour déployer une nouvelle AMI personnalisée mise à jour.

   a. Créez une version de votre AMI.
   b. Créez une version de modèle de lancement avec le nouvel ID d'AMI.
   c. Mettez à niveau les nœuds vers la nouvelle version du modèle de lancement.

4. Dans la boîte de dialogue Update node group version (Mettre à jour la version du groupe de nœuds), activez ou désactivez les options suivantes :
   - Update node group version (Mettre à jour la version du groupe de nœuds) : cette option n'est pas disponible si vous avez déployé une AMI personnalisée ou que votre AMI optimisée pour Amazon EKS correspond actuellement à la dernière version de votre cluster.
   - Change launch template version (Modifier la version du modèle de lancement) : cette option n'est pas disponible si le groupe de nœuds est déployé sans modèle de lancement personnalisé. Vous pouvez uniquement mettre à jour la version du modèle de lancement pour un groupe de nœuds qui a été déployé avec un modèle de lancement personnalisé. Sélectionnez la Version du modèle de lancement vers laquelle vous souhaitez mettre à jour le groupe de nœuds. Si votre groupe de nœuds est configuré avec une AMI personnalisée, la version que vous sélectionnez doit également spécifier une AMI. Lorsque vous passez à une version plus récente de votre modèle de lancement, chaque nœud est recyclé pour correspondre à la nouvelle configuration de la version du modèle de lancement spécifiée.
5. Pour Update strategy (Politique de mise à jour), sélectionnez l'une des options suivantes :
   - Mise à jour continue — Cette option respecte les budgets d'interruption du Pod pour votre cluster. Les mises à jour échouent en cas d'interruption du budget des pods qui empêche Amazon EKS de vider correctement les pods qui s'exécutent sur ce groupe de nœuds.
   - Forcer la mise à jour : cette option ne respecte pas les budgets d'interruption du Pod. Les mises à jour ont lieu indépendamment des problèmes budgétaires liés à l'interruption du pod en forçant le redémarrage des nœuds.
6. Choisissez Mettre à jour.

Modification de la configuration d'un groupe de nœuds

Vous pouvez modifier une partie de la configuration d'un groupe de nœuds gérés.

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster qui contient le groupe de nœuds à modifier.
3. Sélectionnez l'onglet Compute (Calcul).
4. Sélectionnez le groupe de nœuds à modifier, puis choisissez Edit (Modifier).
5. (Facultatif) Sur la page Edit node group (Modifier le groupe de nœuds), effectuez les opérations suivantes :
   a. Modifiez la Node group scaling configuration (Configuration de mise à l'échelle du groupe de nœuds).

- Taille souhaitée : spécifiez le nombre actuel de nœuds que le groupe de nœuds gérés doit conserver au lancement.
- Taille minimale : spécifiez le nombre minimal de nœuds vers lequel le groupe de nœuds gérés peut être mis à l'échelle.
- Taille maximale : spécifiez le nombre maximal de nœuds vers lequel le groupe de nœuds gérés peut être mis à niveau. Pour connaître le nombre maximal de noeuds pris en charge dans un groupe de nœuds, consultez the section called "Quotas de service".

b. (Facultatif) Ajoutez ou supprimez des étiquettes Kubernetes aux nœuds de votre groupe de nœuds. Les labels affichés ici sont uniquement ceux que vous avez appliqués avec Amazon EKS. D'autres étiquettes peuvent exister sur vos nœuds qui ne sont pas affichées ici.

c. (Facultatif) Ajoutez ou supprimez des taches Kubernetes sur les nœuds de votre groupe de nœuds. Les taints ajoutés peuvent avoir l'effet de **`NoSchedule`**, **`NoExecute`** ou **`PreferNoSchedule`**. Pour de plus amples informations, veuillez consulter the section called "Empêcher la planification des pods sur des nœuds spécifiques".

d. (Facultatif) Ajoutez ou supprimez des Tags (Balises) de la ressource de votre groupe de nœuds. Ces identifications ne sont appliquées qu'au groupe de nœuds Amazon EKS. Ils ne se propagent pas à d'autres ressources, telles que les sous-réseaux ou les EC2 instances Amazon du groupe de nœuds.

e. (Facultatif) Modifiez la Configuration des mises à jour du groupe. Sélectionnez Number (Nombre) ou Percentage (Pourcentage).
   - Nombre : sélectionnez et spécifiez le nombre de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle. Ces nœuds ne seront pas disponibles pendant la mise à jour.
   - Pourcentage : sélectionnez et spécifiez le pourcentage de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle. Ces nœuds ne seront pas disponibles pendant la mise à jour. Ceci est utile si vous avez beaucoup de nœuds dans votre groupe de nœuds.

f. Lorsque vous avez terminé les modifications, choisissez Enregistrer les modifications.

> **Important**
>
> Lors de la mise à jour de la configuration du groupe de nœuds, la modification du `NodegroupScalingConfig`ne respecte pas les budgets d'interruption du pod (PDBs). Contrairement au processus de mise à jour des groupes de nœuds (qui draine les nœuds et les respecte PDBs pendant la phase de mise à niveau), la mise à jour de la configuration de dimensionnement entraîne la fermeture immédiate des nœuds via un appel de réduction

automatique du groupe Auto Scaling (ASG). Cela se produit sans réfléchir PDBs, quelle que soit la taille de la cible à laquelle vous réduisez la taille. Cela signifie que lorsque vous réduisez le nombre `desiredSize` d'un groupe de nœuds géré par Amazon EKS, les pods sont expulsés dès que les nœuds sont fermés, sans en honorer aucun. PDBs

Comprenez chaque phase des mises à jour des nœuds

La stratégie de mise à jour des composants master d'Amazon EKS comporte quatre phases différentes décrites dans les sections suivantes.

Phase de configuration

La phase de configuration comporte les étapes suivantes :

1. Il crée une nouvelle version du modèle de EC2 lancement Amazon pour le groupe Auto Scaling associé à votre groupe de nœuds. La nouvelle version du modèle de lancement utilise l'AMI cible ou une version personnalisée du modèle de lancement pour la mise à jour.
2. Il met à jour le groupe Auto Scaling pour utiliser la dernière version du modèle de lancement.
3. La quantité maximale de nœuds à mettre à niveau en parallèle est déterminée à l'aide de la propriété `updateConfig` pour le groupe de nœuds. Le maximum indisponible a un quota de 100 nœuds. La valeur par défaut est de un nœud. Pour plus d'informations, consultez la propriété UpdateConfig dans le manuel Amazon EKS API Reference.

Phase d'augmentation d'échelle

Lors de la mise à niveau des nœuds d'un groupe de nœuds gérés, les nœuds mis à niveau sont lancés dans la même zone de disponibilité que ceux qui sont mis à niveau. Pour garantir ce placement, nous utilisons le rééquilibrage des zones EC2 de disponibilité d'Amazon. Pour plus d'informations, consultez la section Rebalancing des zones de disponibilité dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling. Pour répondre à cette exigence, il est possible que nous lancions jusqu'à deux instances par zone de disponibilité dans votre groupe de nœuds gérés.

La phase d'augmentation d'échelle comporte les étapes suivantes :

1. Il augmente la taille maximale du groupe Auto Scaling et la taille souhaitée de la plus grande des deux valeurs suivantes :

- Jusqu'à deux fois le nombre de zones de disponibilité dans lesquelles le groupe Auto Scaling est déployé.
- Le maximum indisponible de la mise à niveau.

  Par exemple, si votre groupe de nœuds a cinq zones de disponibilité et que `maxUnavailable` est égal à un, le processus de mise à niveau peut lancer un maximum de 10 nœuds. Cependant, lorsqu'il `maxUnavailable` est égal à 20 (ou à une valeur supérieure à 10), le processus lancera 20 nouveaux nœuds.

2. Après la mise à l'échelle du groupe Auto Scaling, elle vérifie si les nœuds utilisant la dernière configuration sont présents dans le groupe de nœuds. Cette étape ne réussit que si elle répond à ces critères :
   - Au moins un nouveau nœud est lancé dans chaque zone de disponibilité où le nœud existe.
   - Chaque nouveau nœud doit être dans l'état `Ready`.
   - Les nouveaux nœuds doivent avoir des étiquettes Amazon EKS appliquées.

     Il s'agit des étiquettes Amazon EKS appliquées sur les composants master d'un groupe de nœuds réguliers :

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`

     Il s'agit des étiquettes appliquées par Amazon EKS sur les composants master dans un modèle de lancement personnalisé ou un groupe de composants AMI :

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`
     - `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`
3. Il marque les nœuds comme non planifiables pour éviter de planifier de nouveaux pods. Il étiquette également les nœuds avec `node.kubernetes.io/exclude-from-external-load-balancers=true` pour supprimer les anciens nœuds des équilibreurs de charge avant de les arrêter.

Les raisons suivantes sont connues pour provoquer une erreur `NodeCreationFailure` dans cette phase :

Capacité insuffisante dans la zone de disponibilité

Il est possible que la zone de disponibilité n'ait pas la capacité des types d'instance demandés. Il est recommandé de configurer plusieurs types d'instances lors de la création d'un groupe de nœuds gérés.

EC2 limites d'instances dans votre compte

Vous devrez peut-être augmenter le nombre d' EC2 instances Amazon que votre compte peut exécuter simultanément à l'aide de Service Quotas. Pour plus d'informations, consultez la section EC2 Service Quotas dans le guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

Données utilisateur personnalisées

Les données utilisateur personnalisées peuvent parfois interrompre le processus d'amorçage. Ce scénario peut conduire à ce que le `kubelet` ne démarre pas sur le nœud ou que les nœuds n'obtiennent pas les étiquettes Amazon EKS attendues. Pour de plus amples informations, veuillez consulter the section called “Spécification d'une AMI”.

Toute modification qui rend un nœud défectueux ou n'est pas prêt

La pression du disque sur le nœud, la pression de la mémoire et des conditions similaires peuvent empêcher un nœud de passer à l'état `Ready`.

Chaque nœud démarre le plus rapidement en 15 minutes

Si un nœud met plus de 15 minutes à démarrer et à rejoindre le cluster, la mise à niveau expirera. Il s'agit du temps d'exécution total pour amorcer un nouveau nœud, mesuré entre le moment où un nouveau nœud est requis et le moment où il rejoint le cluster. Lors de la mise à niveau d'un groupe de nœuds gérés, le compteur de temps démarre dès que la taille du groupe Auto Scaling augmente.

Phase de mise à niveau

La phase de mise à niveau se comporte de deux manières différentes, en fonction de la stratégie de mise à jour. Il existe deux stratégies de mise à jour : par défaut et minimale.

Nous recommandons la stratégie par défaut dans la plupart des scénarios. Il crée de nouveaux nœuds avant de mettre fin aux anciens, afin que la capacité disponible soit maintenue pendant la phase de mise à niveau. La stratégie minimale est utile dans les scénarios où vous êtes limité en termes de ressources ou de coûts, par exemple avec des accélérateurs matériels tels que. GPUs Il

met fin aux anciens nœuds avant d'en créer de nouveaux, de sorte que la capacité totale n'augmente jamais au-delà de la quantité configurée.

La stratégie de mise à jour par défaut comporte les étapes suivantes :

1. Cela augmente le nombre de nœuds (nombre souhaité) dans le groupe Auto Scaling, ce qui oblige le groupe de nœuds à créer des nœuds supplémentaires.
2. Un nœud est sélectionné aléatoirement pour mise à niveau, jusqu'au maximum indisponible configuré pour le groupe de nœuds.
3. Il draine les pods du nœud. Si les Pods ne quittent pas le nœud dans les 15 minutes et qu'aucun indicateur de force n'apparaît, la phase de mise à niveau échoue avec une `PodEvictionFailure` erreur. Dans ce scénario, vous pouvez appliquer l'indicateur de force à la `update-nodegroup-version` demande de suppression des pods.
4. Il boucle le nœud après l'expulsion de chaque pod et attend 60 secondes. Ceci est fait pour que le contrôleur de service n'envoie aucune nouvelle demande à ce nœud et supprime ce nœud de sa liste de nœuds actifs.
5. Une demande d'arrêt est envoyée au groupe Auto Scaling pour le nœud isolé.
6. Les étapes de mise à niveau précédentes sont répétées jusqu'à ce que le groupe de nœuds ne comporte plus aucun nœud déployé avec la version antérieure du modèle de lancement.

La stratégie de mise à jour minimale comprend les étapes suivantes :

1. Il cordonne tous les nœuds du groupe de nœuds au début, de sorte que le contrôleur de service n'envoie aucune nouvelle demande à ces nœuds.
2. Un nœud est sélectionné aléatoirement pour mise à niveau, jusqu'au maximum indisponible configuré pour le groupe de nœuds.
3. Il draine les pods des nœuds sélectionnés. Si les Pods ne quittent pas le nœud dans les 15 minutes et qu'aucun indicateur de force n'apparaît, la phase de mise à niveau échoue avec une `PodEvictionFailure` erreur. Dans ce scénario, vous pouvez appliquer l'indicateur de force à la `update-nodegroup-version` demande de suppression des pods.
4. Une fois que chaque pod est expulsé et attend 60 secondes, il envoie une demande de résiliation au groupe Auto Scaling pour les nœuds sélectionnés. L'Auto Scaling Group crée de nouveaux nœuds (identique au nombre de nœuds sélectionnés) pour remplacer la capacité manquante.
5. Les étapes de mise à niveau précédentes sont répétées jusqu'à ce que le groupe de nœuds ne comporte plus aucun nœud déployé avec la version antérieure du modèle de lancement.

`PodEvictionFailure`erreurs lors de la phase de mise à niveau

Les raisons suivantes sont connues pour provoquer une erreur `PodEvictionFailure` dans cette phase :

PDB agressif

Un PDB agressif est défini sur le Pod ou plusieurs PDBs pointent vers le même Pod.

Déploiement tolérant toutes les souillures

Une fois que chaque pod est expulsé, on s'attend à ce que le nœud soit vide car il a été contaminé lors des étapes précédentes. Toutefois, si le déploiement tolère toutes les souillures, le nœud est plus susceptible de ne pas être vide, ce qui entraînera l'échec de l'expulsion du pod.

Phase de réduction d'échelle

La phase de réduction d'échelle diminue la taille maximale et la taille souhaitée du groupe Auto Scaling d'une unité pour revenir aux valeurs avant le début de la mise à jour.

Si le flux de mise à jour détermine que le Cluster Autoscaler augmente le groupe de nœuds pendant la phase de réduction d'échelle du flux de travail, il se termine immédiatement sans ramener le groupe de nœuds à sa taille d'origine.

Personnalisez les nœuds gérés avec des modèles de lancement

Pour le plus haut niveau de personnalisation, vous pouvez déployer des nœuds gérés en utilisant votre propre modèle de lancement. L'utilisation d'un modèle de lancement permet d'accéder à des fonctionnalités telles que les suivantes :

- Fournissez des arguments bootstrap lors du déploiement d'un nœud, tels que des arguments kubelet supplémentaires.
- Attribuez des adresses IP aux pods à partir d'un bloc CIDR différent de l'adresse IP attribuée au nœud.
- Déployez votre propre AMI personnalisée sur les nœuds.
- Déployez votre propre CNI personnalisée sur les nœuds.

Si vous fournissez votre propre modèle de lancement lors de la création en premier lieu d'un groupe de nœuds gérés, vous bénéficierez également d'une plus grande flexibilité par la suite. Pourvu que

vous déployiez un groupe de nœuds gérés avec votre propre modèle de lancement, vous pouvez le mettre à jour de façon itérative avec une version différente du même modèle de lancement. Lorsque vous mettez à jour votre groupe de nœuds vers une version différente de votre modèle de lancement, tous les nœuds du groupe sont recyclés pour correspondre à la nouvelle configuration de la version spécifiée du modèle de lancement.

Les groupes de nœuds gérés sont toujours déployés avec un modèle de lancement à utiliser avec le groupe Amazon EC2 Auto Scaling. Lorsque vous ne fournissez pas de modèle de lancement, l'API Amazon EKS en crée un automatiquement avec des valeurs par défaut dans votre compte. Toutefois, nous vous déconseillons de modifier les modèles de lancement générés automatiquement. En outre, les groupes de nœuds existants qui n'utilisent pas de modèle de lancement personnalisé ne peuvent pas être mis à jour directement. Vous devez plutôt créer un nouveau groupe de nœuds avec un modèle de lancement personnalisé à cet effet.

Concepts de base de la configuration d'un modèle de lancement

Vous pouvez créer un modèle de lancement Amazon EC2 Auto Scaling à l' AWS Management Console aide de la AWS CLI ou d'un AWS SDK. Pour plus d'informations, consultez la section Création d'un modèle de lancement pour un groupe Auto Scaling dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling. Certains des paramètres d'un modèle de lancement sont similaires aux paramètres utilisés pour la configuration des nœuds gérés. Lors du déploiement ou de la mise à jour d'un groupe de nœuds avec un modèle de lancement, certains paramètres doivent être spécifiés soit dans la configuration du groupe de nœuds, soit dans le modèle de lancement. Ne spécifiez pas de paramètre aux deux endroits. Si un paramètre existe là où il ne devrait pas exister, les opérations telles que la création ou la mise à jour d'un groupe de nœuds échouent.

Le tableau suivant répertorie les paramètres interdits dans un modèle de lancement. Il répertorie également les paramètres similaires, s'il en existe, qui sont requis dans la configuration du groupe de nœuds géré. Les paramètres répertoriés sont les paramètres qui apparaissent dans la console. Ils peuvent avoir des noms similaires mais différents dans la AWS CLI et le SDK.

Modèle de lancement – Interdit	Configuration d'un groupe de nœuds Amazon EKS
Sous-réseau sous Interfaces réseau (Ajouter une interface réseau)	Sous-réseaux sous Configuration réseau du groupe de nœuds sur la page Spécifier le réseau

Modèle de lancement – Interdit	Configuration d'un groupe de nœuds Amazon EKS
Profil d'instance IAM sous Détails avancés	Rôle IAM de nœud sous Configuration du groupe de nœuds sur la page Configurer le groupe de nœuds
Comportement d'arrêt et Arrêt – Comportement de mise en veille prolongée sous Détails avancés. Conserver la valeur par défaut Ne pas inclure dans le paramètre du modèle de lancement dans le modèle de lancement pour les deux paramètres.	Pas d'équivalent. Amazon EKS doit contrôler le cycle de vie de l'instance et non pas le groupe Auto Scaling.

Le tableau suivant répertorie les paramètres interdits dans une configuration de groupe de nœuds gérée. Il répertorie également les paramètres similaires, s'il en existe, qui sont requis dans un modèle de lancement. Les paramètres répertoriés sont les paramètres qui apparaissent dans la console. Ils peuvent porter des noms similaires dans la AWS CLI et le SDK.

Configuration du groupe de nœuds Amazon EKS : Interdite	Modèle de lancement
(Seulement si vous avez spécifié une AMI personnalisée dans un modèle de lancement) AMI type (Type d'AMI) sous Node Group compute configuration (Configuration du calcul du groupe de nœuds) sur la page Set compute and scaling configuration (Définir la configuration de calcul et de mise à l'échelle) : la console affiche Specified in launch template (Spécifié dans le modèle de lancement) et l'ID d'AMI spécifié. Si les images de l'application et du système d'exploitation (Amazon Machine Image) n'ont pas été spécifiées dans le modèle de	Images d'applications et de systèmes d'exploitation (Amazon Machine Image) sous Contenu du modèle de lancement : vous devez spécifier un ID dans l'un des cas suivants : • Utilisation d'une image AMI personnalisée Si vous spécifiez une AMI qui ne répond pas aux exigences répertoriées dans Spécification d'une AMI, le déploiement du groupe de nœuds échouera. • Veut fournir des données utilisateur pour fournir des arguments au fichier `bootstrap.sh` inclus avec un AMI optimisé pour Amazon EKS. Vous pouvez permettre à vos

Configuration du groupe de nœuds Amazon EKS : Interdite	Modèle de lancement
lancement, vous pouvez sélectionner une AMI dans la configuration du groupe de nœuds.	instances d'attribuer un nombre nettement plus élevé d'adresses IP aux pods, d'attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de celui de l'instance ou de déployer un cluster privé sans accès Internet sortant. Pour plus d'informations, consultez les rubriques suivantes : • Attribuez davantage d'adresses IP aux nœuds Amazon EKS avec des préfixes • Déployez des pods dans des sous-réseaux alternatifs avec un réseau personnalisé • Déployez des clusters privés avec un accès Internet limité • Spécification d'une AMI
Taille du disque sous Configuration de calcul du groupe de nœuds sur la page Définir la configuration de calcul et de mise à l'échelle : la console affiche Spécifié dans le modèle de lancement.	Taille sous Stockage (volumes) (Ajouter un nouveau volume). Vous devez le spécifier dans le modèle de lancement.
Paire de clés SSH sous Configuration du groupe de nœuds sur la page Spécifier le réseau : la console affiche la clé spécifiée dans le modèle de lancement ou Non spécifié dans le modèle de lancement.	Nom de la paire de clés sous Paire de clés (connexion).
Vous ne pouvez pas spécifier les groupes de sécurité source autorisés à accéder à distance lorsque vous utilisez un modèle de lancement.	Groupes de sécurité sous Paramètres réseau pour l'instance ou Groupes de sécurité sous Interfaces réseau (Ajouter une interface réseau), mais pas les deux. Pour de plus amples informations, veuillez consulter the section called "Utilisation des groupes de sécurité".

Note

- Si vous déployez un groupe de nœuds à l'aide d'un modèle de lancement, spécifiez zéro ou un type d'instance sous Contenu du modèle de lancement dans un modèle de lancement. Vous pouvez également spécifier de 0 à 20 types d'instance sous types d'instance sur la page Définir la configuration de calcul et de mise à l'échelle dans la console. Vous pouvez également le faire à l'aide d'autres outils qui utilisent l'API Amazon EKS. Si vous spécifiez un type d'instance dans un modèle de lancement et que vous utilisez ce modèle de lancement pour déployer votre groupe de nœuds, vous ne pouvez pas spécifier de type d'instance dans la console ou à l'aide d'autres outils utilisant l'API Amazon EKS. Si vous ne spécifiez aucun type d'instance dans un modèle de lancement, dans la console ou à l'aide d'autres outils utilisant l'API Amazon EKS, le type d'`t3.medium`instance est utilisé. Si votre groupe de nœuds utilise le type de capacité Spot, nous vous recommandons de spécifier plusieurs types d'instance à l'aide de la console. Pour de plus amples informations, veuillez consulter the section called "Types de capacité des groupes de nœuds gérés".
- Si les conteneurs que vous déployez dans le groupe de nœuds utilisent le service de métadonnées d'instance version 2, veillez à définir 2 comme limite de saut de réponse des métadonnées dans votre modèle de lancement. Pour plus d'informations, consultez la section Métadonnées de l'instance et données utilisateur dans le guide de EC2 l'utilisateur Amazon.
- Les modèles de lancement ne prennent pas en charge la `InstanceRequirements` fonctionnalité qui permet une sélection flexible du type d'instance.

Marquage des instances Amazon EC2

Vous pouvez utiliser le `TagSpecification` paramètre d'un modèle de lancement pour spécifier les balises à appliquer aux EC2 instances Amazon de votre groupe de nœuds. L'entité IAM qui appelle le `CreateNodegroup` ou `UpdateNodegroupVersion` APIs doit disposer des autorisations pour `ec2:RunInstances` et`ec2:CreateTags`, et les balises doivent être ajoutées au modèle de lancement.

Utilisation des groupes de sécurité

Vous pouvez utiliser un modèle de lancement pour spécifier les groupes EC2 de sécurité Amazon personnalisés à appliquer aux instances de votre groupe de nœuds. Vous pouvez le faire soit dans

le paramètre des groupes de sécurité au niveau de l'instance, soit dans le cadre des paramètres de configuration de l'interface réseau. Toutefois, vous ne pouvez pas créer un modèle de lancement qui spécifie à la fois le niveau de l'instance et les groupes de sécurité de l'interface réseau. Prenez en compte les conditions suivantes qui s'appliquent à l'utilisation de groupes de sécurité personnalisés avec des groupes de nœuds gérés :

- Lorsque vous utilisez le AWS Management Console, Amazon EKS n'autorise que les modèles de lancement dotés d'une seule spécification d'interface réseau.
- Par défaut, Amazon EKS applique le groupe de sécurité du cluster aux instances de votre groupe de nœuds pour faciliter la communication entre les nœuds et le plan de contrôle. Si vous spécifiez des groupes de sécurité personnalisés dans le modèle de lancement à l'aide de l'une des options mentionnées précédemment, Amazon EKS n'ajoute pas le groupe de sécurité du cluster. Vous devez donc vous assurer que les règles d'entrée et de sortie de vos groupes de sécurité permettent la communication avec le point de terminaison de votre cluster. Si les règles de votre groupe de sécurité sont incorrectes, les nœuds de travail ne peuvent pas rejoindre le cluster. Pour plus d'informations sur les règles de groupe de sécurité, consultez the section called “Exigences de groupe de sécurité”.
- Si vous avez besoin d'un accès SSH aux instances de votre groupe de nœuds, incluez un groupe de sécurité qui autorise cet accès.

Données EC2 utilisateur d'Amazon

Le modèle de lancement comprend une section pour les données utilisateur personnalisées. Vous pouvez définir les paramètres de configuration de votre groupe de nœuds dans cette section sans créer manuellement une personnalisation individuelle AMIs. Pour plus d'informations sur les paramètres disponibles pour Bottlerocket, consultez la section Utilisation des données utilisateur sur. GitHub

Vous pouvez fournir des données EC2 utilisateur Amazon dans votre modèle de lancement `cloud-init` lorsque vous lancez vos instances. Pour plus d'informations, consultez la documentation cloud-init. Vos données utilisateur peuvent être utilisées pour effectuer des opérations de configuration courantes. Elles comprennent :

- Inclusion d'utilisateurs ou de groupes
- Installations de packages

Les données EC2 utilisateur Amazon figurant dans les modèles de lancement utilisés avec des groupes de nœuds gérés doivent être au format d'archive en plusieurs parties MIME pour Amazon Linux AMIs et au format TOML pour Bottlerocket. AMIs En effet, vos données utilisateur sont fusionnées avec les données utilisateur d'Amazon EKS requises pour que les nœuds puissent rejoindre le cluster. Ne spécifiez aucune commande démarrant ou modifiant `kubelet` dans vos données utilisateur. Ceci est effectué dans le cadre des données utilisateur fusionnées par Amazon EKS. Certains paramètres `kubelet`, tels que la définition des étiquettes sur les nœuds, peuvent être configurés directement via l'API des groupes de nœuds gérés.

> **Note**
>
> Pour plus d'informations sur la personnalisation avancée de `kubelet`, y compris son démarrage manuel ou le passage de paramètres de configuration personnalisés, consultez the section called "Spécification d'une AMI". Si un ID AMI personnalisé est spécifié dans un modèle de lancement, Amazon EKS ne fusionne pas les données utilisateur.

Les détails suivants fournissent plus d'informations sur la section des données utilisateur.

Données utilisateur d'Amazon Linux 2

Vous pouvez combiner plusieurs blocs de données utilisateur dans un seul fichier MIME multi-part. Par exemple, vous pouvez combiner un boothook cloud qui configure le démon Docker avec un script shell de données utilisateur qui installe un package personnalisé. Un fichier MIME multi-part est constitué des composants suivants :

- La déclaration de type de contenu et de limite de partie : `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- La déclaration de version MIME : `MIME-Version: 1.0`
- Un ou plusieurs blocs de données qui contiennent les composants suivants :
  - La limite de début qui signale le début d'un bloc de données utilisateur : `--==MYBOUNDARY==`
  - La déclaration de type de contenu du bloc : `Content-Type: text/cloud-config; charset="us-ascii"`. Pour plus d'informations sur les types de contenus, consultez la documentation sur Cloud-Init.
  - Le contenu des données utilisateur (par exemple, une liste de commandes shell ou de directives `cloud-init`).

- La limite de fin qui signale la fin du fichier MIME multi-part : `--==MYBOUNDARY==--`

Voici un exemple de fichier MIME multi-part que vous pouvez utiliser pour créer le vôtre.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Données utilisateur d'Amazon Linux 2023

Amazon Linux 2023 (AL2023) introduit un nouveau processus d'initialisation des nœuds `nodeadm` qui utilise un schéma de configuration YAML. Si vous utilisez des groupes de nœuds autogérés ou une AMI avec un modèle de lancement, vous devez désormais fournir des métadonnées de cluster supplémentaires de manière explicite lors de la création d'un nouveau groupe de nœuds. Voici un exemple des paramètres minimaux requis, où `apiServerEndpointcertificateAuthority`, et le service `cidr` sont désormais requis :

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Vous définissez généralement cette configuration dans vos données utilisateur, telle quelle ou intégrée dans un document MIME en plusieurs parties :

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
```

```
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

Dans AL2, les métadonnées de ces paramètres ont été découvertes à partir de l'appel d'`DescribeCluster`API Amazon EKS. Avec AL2 023, ce comportement a changé car l'appel d'API supplémentaire risque d'être limité lors de la mise à l'échelle de nœuds à grande échelle. Cette modification ne vous concerne pas si vous utilisez des groupes de nœuds gérés sans modèle de lancement ou si vous utilisez Karpenter. Pour plus d'informations sur les services `certificateAuthority` et les servicescidr, consultez `DescribeCluster`le manuel Amazon EKS API Reference.

Voici un exemple complet de données utilisateur AL2 023 qui combine un script shell pour personnaliser le nœud (comme l'installation de packages ou la pré-mise en cache des images de conteneur) avec la configuration requise. `nodeadm` Cet exemple montre les personnalisations courantes, notamment :* Installation de packages système supplémentaires * Configuration de la pré-mise en cache des images de conteneur pour améliorer le temps de démarrage du pod * Configuration de la configuration du proxy HTTP * Configuration des `kubelet` indicateurs pour l'étiquetage des nœuds

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -o errexit
set -o pipefail
set -o nounset

# Install additional packages
yum install -y htop jq iptables-services

# Pre-cache commonly used container images
nohup docker pull public.ecr.aws/eks-distro/kubernetes/pause:3.2 &
```

```
# Configure HTTP proxy if needed
cat > /etc/profile.d/http-proxy.sh << 'EOF'
export HTTP_PROXY="http://proxy.example.com:3128"
export HTTPS_PROXY="http://proxy.example.com:3128"
export NO_PROXY="localhost,127.0.0.1,169.254.169.254,.internal"
EOF

--BOUNDARY
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      clusterDNS:
      - 10.100.0.10
    flags:
    - --node-labels=app=my-app,environment=production

--BOUNDARY--
```

Données utilisateur de Bottlerocket

Bottlerocket structure les données utilisateur au format TOML. Vous pouvez fournir des données utilisateur qui seront fusionnées avec les données utilisateur fournies par Amazon EKS. Par exemple, vous pouvez fournir des paramètres `kubelet` supplémentaires.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Pour plus d'informations sur les paramètres pris en charge, consultez la documentation Bottlerocket. Vous pouvez configurer des étiquettes de nœuds et des rejets dans vos données utilisateur. Cependant, nous vous recommandons de les configurer plutôt dans votre groupe de nœuds. Amazon EKS applique ces configurations lorsque vous le faites.

Lorsque les données utilisateur sont fusionnées, le formatage n'est pas préservé, mais le contenu reste le même. La configuration que vous fournissez dans vos données utilisateur remplace tous les paramètres qui sont configurés par Amazon EKS. Ainsi, si vous définissez `settings.kubernetes.max-pods` ou`settings.kubernetes.cluster-dns-ip`, ces valeurs de vos données utilisateur sont appliquées aux nœuds.

Amazon EKS ne prend pas en charge tous les fichiers TOML valides. Voici une liste des formats connus non pris en charge :

- Guillemets dans les clés entre guillemets : `'quoted "value"' = "value"`
- Guillemets échappés dans les valeurs : `str = "I'm a string. \"You can quote me \""`
- Flottants et entiers mélangés : `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- Types mixtes dans les tableaux : `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- En-têtes entre parenthèses avec clés entre guillemets : `[foo."bar.baz"]`

Données utilisateur Windows

Les données utilisateur Windows utilisent des PowerShell commandes. Lorsque vous créez un groupe de nœuds gérés, vos données utilisateur personnalisées sont combinées avec les données utilisateur gérées par Amazon EKS. Vos PowerShell commandes viennent en premier, suivies des commandes de gestion des données utilisateur, le tout dans une seule `<powershell></powershell>` balise.

> **Important**
>
> Lors de la création de groupes de nœuds Windows, Amazon EKS les met `aws-auth ConfigMap` à jour pour permettre aux nœuds basés sur Linux de rejoindre le cluster. Le service ne configure pas automatiquement les autorisations pour Windows AMIs. Si vous utilisez des nœuds Windows, vous devez gérer l'accès via l'API d'entrée d'accès ou en la mettant à jour `aws-auth ConfigMap` directement. Pour de plus amples informations, veuillez consulter the section called "Activer le support Windows".

> **Note**
>
> Lorsqu'aucun ID AMI n'est spécifié dans le modèle de lancement, n'utilisez pas le script Windows Amazon EKS Bootstrap dans les données utilisateur pour configurer Amazon EKS.

Voici un exemple de données utilisateur.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

Spécification d'une AMI

Si vous avez l'une des conditions suivantes, spécifiez un ID d'AMI dans le champ `ImageId` de votre modèle de lancement. Sélectionnez votre besoin d'informations supplémentaires.

Fournissez des données utilisateur pour transmettre des arguments au **`bootstrap.sh`** fichier inclus dans une Linux/Bottlerocket AMI optimisée pour Amazon EKS

L'amorçage est un terme utilisé pour décrire l'ajout de commandes pouvant être exécutées au démarrage d'une instance. Par exemple, le bootstrapping permet d'utiliser des arguments kubelet supplémentaires. Vous pouvez transmettre les arguments au script `bootstrap.sh` en utilisant `eksctl` sans spécifier de modèle de lancement. Vous pouvez également le faire en spécifiant les informations dans la section des données utilisateur d'un modèle de lancement.

eksctl sans spécifier de modèle de lancement

Créez un fichier nommé *`my-nodegroup.yaml`* avec les contenus suivants. Remplacez chaque *`example value`* par vos propres valeurs. Les arguments `--apiserver-endpoint`, `--b64-cluster-ca` et `--dns-cluster-ip` sont facultatifs. Cependant, leur définition permet au script `bootstrap.sh` d'éviter de passer un appel `describeCluster`. Cela est utile dans les configurations de clusters privés ou dans les clusters où vous augmentez fréquemment le nombre de nœuds entrants et sortants. Pour plus d'informations sur le `bootstrap.sh` script, consultez le fichier bootstrap.sh sur GitHub.

- Le seul argument requis est le nom du cluster (*`my-cluster`*).

- Pour récupérer un ID d'AMI optimisé pour`ami-`*`1234567890abcdef0`* , consultez les sections suivantes :
  - Récupérez l'AMI Amazon Linux recommandée IDs
  - Récupérez l'AMI Bottlerocket recommandée IDs
  - Récupérez l'AMI Microsoft Windows recommandée IDs
- Pour récupérer le *`certificate-authority`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Pour récupérer le *`api-server-endpoint`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- La valeur de `--dns-cluster-ip` est votre CIDR de service avec `.10` à la fin. Pour récupérer le *`service-cidr`* de votre cluster, exécutez la commande suivante. Par exemple, si la valeur renvoyée est `ipv4 10.100.0.0/16`, votre valeur est *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Cet exemple fournit un argument `kubelet` supplémentaire pour définir une valeur `max-pods` personnalisée à l'aide du script `bootstrap.sh` inclus dans l'AMI optimisée pour Amazon EKS. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Pour obtenir de l'aide sur la sélection de *`my-max-pods-value`*, consultez the section called “Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon”.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
```

```
managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

Pour chaque option de fichier `eksctl config` disponible, consultez Schéma de fichier de configuration dans la documentation `eksctl`. L'utilitaire `eksctl` crée toujours un modèle de lancement pour vous et remplit ses données utilisateur avec les données que vous fournissez dans le fichier `config`.

Créez un groupe de nœuds avec la commande suivante.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

Données utilisateur dans un modèle de lancement

Spécifiez les informations suivantes dans la section des données utilisateur de votre modèle de lancement. Remplacez chaque *`example value`* par vos propres valeurs. Les arguments `--apiserver-endpoint`, `--b64-cluster-ca` et `--dns-cluster-ip` sont facultatifs. Cependant, leur définition permet au script `bootstrap.sh` d'éviter de passer un appel `describeCluster`. Cela est utile dans les configurations de clusters privés ou dans les clusters où vous augmentez fréquemment le nombre de nœuds entrants et sortants. Pour plus d'informations sur le `bootstrap.sh` script, consultez le fichier bootstrap.sh sur GitHub.

- Le seul argument requis est le nom du cluster (*`my-cluster`*).
- Pour récupérer le *`certificate-authority`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Pour récupérer le *api-server-endpoint* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- La valeur de `--dns-cluster-ip` est votre CIDR de service avec `.10` à la fin. Pour récupérer le *service-cidr* de votre cluster, exécutez la commande suivante. Par exemple, si la valeur renvoyée est `ipv4 10.100.0.0/16`, votre valeur est *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Cet exemple fournit un argument `kubelet` supplémentaire pour définir une valeur `max-pods` personnalisée à l'aide du script `bootstrap.sh` inclus dans l'AMI optimisée pour Amazon EKS. Pour obtenir de l'aide sur la sélection de *my-max-pods-value*, consultez the section called "Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon".

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Fournissez des données utilisateur pour transmettre des arguments au **`Start-EKSBootstrap.ps1`** fichier inclus dans une AMI Windows optimisée pour Amazon EKS

L'amorçage est un terme utilisé pour décrire l'ajout de commandes pouvant être exécutées au démarrage d'une instance. Vous pouvez transmettre les arguments au script `Start-`

`EKSBootstrap.ps1` en utilisant `eksctl` sans spécifier de modèle de lancement. Vous pouvez également le faire en spécifiant les informations dans la section des données utilisateur d'un modèle de lancement.

Si vous souhaitez spécifier un ID AMI Windows personnalisé, tenez compte des considérations suivantes :

- Vous devez utiliser un modèle de lancement et fournir les commandes d'amorçage requises dans la section des données utilisateur. Pour récupérer l'ID Windows souhaité, vous pouvez utiliser le tableau de la section Créer des nœuds avec Windows optimisé AMIs.
- Il existe plusieurs limites et conditions. Par exemple, vous devez ajouter des éléments `eks:kube-proxy-windows` à votre carte de configuration AWS IAM Authenticator. Pour de plus amples informations, veuillez consulter the section called "Limites et conditions lors de la spécification d'un ID d'AMI".

Spécifiez les informations suivantes dans la section des données utilisateur de votre modèle de lancement. Remplacez chaque *`example value`* par vos propres valeurs. Les arguments `-APIServerEndpoint`, `-Base64ClusterCA` et `-DNSClusterIP` sont facultatifs. Cependant, leur définition permet au script `Start-EKSBootstrap.ps1` d'éviter de passer un appel `describeCluster`.

- Le seul argument requis est le nom du cluster (*`my-cluster`*).
- Pour récupérer le *`certificate-authority`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- Pour récupérer le *`api-server-endpoint`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- La valeur de `--dns-cluster-ip` est votre CIDR de service avec `.10` à la fin. Pour récupérer le *`service-cidr`* de votre cluster, exécutez la commande suivante. Par exemple, si la valeur renvoyée est `ipv4 10.100.0.0/16`, votre valeur est *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- Pour des arguments supplémentaires, consultez the section called "Paramètres de configuration du script d'amorçage".

> **Note**
>
> Si vous utilisez un service CIDR personnalisé, vous devez le spécifier à l'aide du `-ServiceCIDR` paramètre. Dans le cas contraire, la résolution DNS des pods du cluster échouera.

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

Exécution d'une AMI personnalisée en raison d'exigences spécifiques en matière de sécurité, de conformité ou de politique interne.

Pour plus d'informations, consultez Amazon Machine Images (AMI) dans le guide de EC2 l'utilisateur Amazon. La spécification de génération de l'AMI Amazon EKS contient des ressources et des scripts de configuration pour créer une AMI Amazon EKS personnalisée basée sur Amazon Linux. Pour plus d'informations, consultez Spécifications de création d'AMI Amazon EKS sur GitHub. Pour créer une AMIs installation personnalisée avec d'autres systèmes d'exploitation, consultez Amazon EKS Sample Custom AMIs on GitHub.

Vous ne pouvez pas utiliser de références de paramètres dynamiques pour les AMI IDs dans les modèles de lancement utilisés avec les groupes de nœuds gérés.

> **Important**
>
> Lorsque vous spécifiez une AMI, Amazon EKS ne fusionne aucune donnée utilisateur. Vous êtes plutôt chargé de fournir les `bootstrap` commandes requises pour que les nœuds rejoignent le cluster. Si vos nœuds ne parviennent pas à joindre le cluster, les actions `CreateNodegroup` et `UpdateNodegroupVersion` Amazon EKS échouent également.

Limites et conditions lors de la spécification d'un ID d'AMI

Voici les limites et les conditions impliquées dans la spécification d'un ID d'AMI avec des groupes de nœuds gérés :

- Vous devez créer un groupe de nœuds pour passer de la spécification d'un ID d'AMI dans un modèle de lancement à la non-spécification d'un ID d'AMI.
- La console ne vous avertit pas lorsqu'une nouvelle version de l'AMI est disponible. Pour mettre à jour votre groupe de nœuds vers une version d'AMI plus récente, vous devez créer une nouvelle version de votre modèle de lancement avec un ID d'AMI mis à jour. Vous devez ensuite mettre à jour le groupe de nœuds avec la nouvelle version du modèle de lancement.
- Les champs suivants ne peuvent pas être définis dans l'API si vous spécifiez un ID d'AMI :
  - `amiType`
  - `releaseVersion`
  - `version`
- Tous les `taints` définis dans l'API sont appliqués de manière asynchrone si vous spécifiez un identifiant d'AMI. Pour appliquer des rejets avant l'ajout d'un nœud au cluster, vous devez les transférer à `kubelet` dans vos données utilisateur à l'aide de l'indicateur de ligne de commande `--register-with-taints`. Pour plus d'informations, consultez kubelet dans la documentation Kubernetes.
- Lorsque vous spécifiez un ID AMI personnalisé pour les groupes de nœuds gérés par Windows, ajoutez-le `eks:kube-proxy-windows` à votre carte de configuration AWS IAM Authenticator. Cela est nécessaire pour le bon fonctionnement du DNS.
  a. Ouvrez la carte de configuration de l'authentificateur AWS IAM pour la modifier.

     ```
     kubectl edit -n kube-system cm aws-auth
     ```

  b. Ajoutez cette entrée à la `groups` liste sous chacun des nœuds Windows `rolearn` associés. Votre carte de configuration doit ressembler à aws-auth-cm-windows.yaml.

     ```
     - eks:kube-proxy-windows
     ```

  c. Enregistrez le fichier et quittez votre éditeur de texte.
- Pour toute AMI utilisant un modèle de lancement personnalisé, la valeur par défaut `HttpPutResponseHopLimit` pour les groupes de nœuds gérés est définie sur2.

Supprimer un groupe de nœuds gérés de votre cluster

Cette rubrique explique comment supprimer un groupe de nœuds gérés Amazon EKS. Lorsque vous supprimez un groupe de nœuds gérés, Amazon EKS définit d'abord 0 pour les tailles minimale, maximale et souhaitée de votre groupe Auto Scaling. Cela entraîne ensuite une réduction d'échelle de votre groupe de nœuds.

Avant la mise hors service de chaque instance, Amazon EKS envoie un signal pour vider les pods de ce nœud. Si les pods ne sont pas épuisés au bout de quelques minutes, Amazon EKS laisse Auto Scaling poursuivre la mise hors service de l'instance. Après la résiliation de chaque instance, le groupe Auto Scaling est supprimé.

> **Important**
>
> Si vous supprimez un groupe de nœuds gérés qui utilise un rôle de nœud IAM qui n'est utilisé par aucun autre groupe de nœuds gérés du cluster, le rôle est supprimé du `aws-authConfigMap`. Si l'un des groupes de nœuds autogérés dans le cluster utilisent le même rôle IAM de nœud, le statut des nœuds autogérés devient `NotReady`. En outre, le fonctionnement du cluster est également perturbé. Pour ajouter un mappage pour le rôle que vous utilisez uniquement pour les groupes de nœuds autogérésthe section called "Création d'entrées d'accès", vérifiez si la version de la plate-forme de votre cluster est au moins la version minimale répertoriée dans la section Conditions préalables de la section Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS. Si la version de votre plateforme est antérieure à la version minimale requise pour les entrées d'accès, vous pouvez réajouter l'entrée dans le `aws-authConfigMap`. Pour plus d'informations, entrez `eksctl create iamidentitymapping --help` dans votre terminal.

Vous pouvez supprimer un groupe de nœuds gérés avec :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

`eksctl`

Supprimer un groupe de nœuds gérés avec **`eksctl`**

Entrez la commande suivante. Remplacez chaque *example value* par vos propres valeurs.

```
eksctl delete nodegroup \
  --cluster my-cluster \
  --name my-mng \
  --region region-code
```

Pour plus d'options, consultez la section Supprimer et vider des groupes de nœuds dans la documentation `eksctl`.

AWS Management Console

Supprimer un groupe de nœuds gérés avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Sur la page Clusters, choisissez le cluster qui contient le groupe de nœuds à supprimer.
3. Sur la page du cluster sélectionné, choisissez l'onglet Compute.
4. Dans la section Node Groups (Groupes de nœuds), choisissez le groupe de nœuds à supprimer. Ensuite, choisissez Supprimer.
5. Dans la boîte de dialogue de confirmation de la suppression du groupe de nœuds, entrez le nom du groupe de nœuds. Ensuite, choisissez Supprimer.

AWS CLI

Supprimer un groupe de nœuds gérés à l'aide de la AWS CLI

1. Entrez la commande suivante. Remplacez chaque *example value* par vos propres valeurs.

   ```
   aws eks delete-nodegroup \
     --cluster-name my-cluster \
     --nodegroup-name my-mng \
     --region region-code
   ```

2. Utilisez les touches fléchées de votre clavier pour faire défiler la sortie de réponse. Appuyez sur la q touche lorsque vous avez terminé.

   Pour plus d'options, consultez la `delete-nodegroup` commande dans le manuel de référence des commandes de la AWS CLI.

Gérez vous-même les nœuds grâce à des nœuds autogérés

Un cluster contient un ou plusieurs EC2 nœuds Amazon sur lesquels les pods sont planifiés. Les nœuds Amazon EKS s'exécutent dans votre AWS compte et se connectent au plan de contrôle de votre cluster via le point de terminaison du serveur API du cluster. Ils vous sont facturés sur la base des EC2 prix Amazon. Pour plus d'informations, consultez les EC2 tarifs Amazon.

Un cluster peut contenir plusieurs groupes de nœuds. Chaque groupe de nœuds contient un ou plusieurs nœuds déployés dans un groupe Amazon EC2 Auto Scaling. Le type d'instance des nœuds au sein du groupe peut varier, par exemple lors de l'utilisation de la sélection du type d'instance basée sur les attributs avec Karpenter. Toutes les instances d'un groupe de nœuds doivent utiliser le rôle IAM du nœud Amazon EKS.

Amazon EKS fournit des images Amazon Machine (AMIs) spécialisées appelées Amazon EKS optimisées AMIs. Ils AMIs sont configurés pour fonctionner avec Amazon EKS. Leurs composants incluent`containerd`,`kubelet`, et l'authentificateur AWS IAM. Ils contiennent AMIs également un script d'amorçage spécialisé qui lui permet de découvrir et de se connecter automatiquement au plan de contrôle de votre cluster.

Si vous limitez l'accès au point de terminaison public de votre cluster à l'aide de blocs CIDR, nous vous recommandons d'activer également l'accès au point de terminaison privé. Cela permet aux nœuds de communiquer avec le cluster. Sans l'activation du point de terminaison privé, les blocs CIDR que vous spécifiez pour l'accès public doivent inclure les sources de sortie de votre VPC. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

Pour ajouter des nœuds autogérés à votre cluster Amazon EKS, consultez les rubriques qui suivent. Si vous lancez des nœuds autogérés manuellement, ajoutez la balise suivante à chaque nœud tout en vous assurant qu'elle `<cluster-name>` correspond à votre cluster. Pour de plus amples informations, veuillez consulter Ajout et suppression de balises sur une ressource individuelle. Si vous suivez les étapes des guides qui suivent, l'identification requise est ajoutée au nœud pour vous.

Clé	Valeur
`kubernetes.io/cluster/<cluster-name>`	`owned`

> **Important**
>
> Les balises d'Amazon EC2 Instance Metadata Service (IMDS) ne sont pas compatibles avec les nœuds EKS. Lorsque les balises de métadonnées d'instance sont activées, l'utilisation de barres obliques («/») dans les valeurs des balises est empêchée. Cette limitation peut entraîner des échecs de lancement d'instances, en particulier lors de l'utilisation d'outils de gestion de nœuds tels que Karpenter ou Cluster Autoscaler, car ces services s'appuient sur des balises contenant des barres obliques pour garantir leur bon fonctionnement.

Pour plus d'informations sur les nœuds à partir d'une perspective Kubernetes générale, consultez Nœuds dans la documentation Kubernetes.

Rubriques

- Créez des nœuds Amazon Linux autogérés
- Créez des nœuds Bottlerocket autogérés
- Créez des nœuds Microsoft Windows autogérés
- Créez des nœuds Ubuntu Linux autogérés
- Mettez à jour les nœuds autogérés pour votre cluster

Créez des nœuds Amazon Linux autogérés

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Linux qui s'enregistrent auprès de votre cluster Amazon EKS. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes. Vous pouvez également lancer des nœuds Amazon Linux autogérés avec `eksctl` ou le AWS Management Console. Si vous devez lancer des nœuds sur AWS Outposts, consultez. the section called "Nœuds"

- Un cluster Amazon EKS existant. Pour en déployer un, consultez the section called "Créer un cluster ". Si vous avez des sous-réseaux dans la AWS région où AWS Outposts, AWS Wavelength ou Local AWS Zones est activé, ces sous-réseaux ne doivent pas avoir été transmis lorsque vous avez créé votre cluster.
- Un rôle IAM existant pour les nœuds à utiliser. Pour en créer un, consultez the section called "Rôle IAM de nœud". Si ce rôle ne possède aucune des politiques du VPC CNI, le rôle distinct qui suit est requis pour les pods CNI du VPC.

- (Facultatif, mais recommandé) Le module complémentaire Amazon VPC CNI pour Kubernetes est configuré avec son propre rôle IAM auquel est attachée la politique IAM nécessaire. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
- Connaissance des considérations répertoriées dans Choisissez un type d'instance de EC2 nœud Amazon optimal. Selon le type d'instance que vous choisissez, il peut y avoir des prérequis supplémentaires pour votre cluster et votre VPC.

Vous pouvez lancer des nœuds Linux autogérés en utilisant l'une des méthodes suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Lancez des nœuds Linux autogérés à l'aide de **`eksctl`**

1. Installez la version `0.210.0` ou une version ultérieure de l'outil de ligne de `eksctl` commande installé sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy est attachée au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
3. La commande suivante crée un groupe de nœuds dans un cluster existant. Remplacer *`al-nodes`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacez les valeurs de *`example value`* restantes par vos propres valeurs. Par défaut, les nœuds sont créés avec la même version de Kubernetes que le plan de contrôle.

   Avant de choisir une valeur pour`--node-type`, consultez l'article Choisissez un type d'instance de EC2 nœud Amazon optimal.

*my-key*Remplacez-le par le nom de votre paire de EC2 clés Amazon ou de votre clé publique. Cette clé est utilisée pour SSH dans vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

Créez votre groupe de nœuds avec la commande suivante.

> **Important**
>
> Si vous souhaitez déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, Wavelength ou Local Zone, vous devez prendre en compte d'autres considérations :
>
> - Les sous-réseaux ne doivent pas avoir été renseignés lors de la création du cluster.
> - Vous devez créer le groupe de nœuds avec un fichier config qui spécifie les sous-réseaux et `volumeType: gp2`. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier config et Schéma du fichier Config dans la documentation `eksctl`.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

Pour déployer un groupe de nœuds qui :

- peut attribuer un nombre d'adresses IP nettement supérieur aux pods par rapport à la configuration par défaut, voirthe section called "Augmenter le nombre d'adresses IP".
- peut attribuer des `IPv4` adresses aux pods à partir d'un bloc CIDR différent de celui de l'instance, voirthe section called "Mise en réseau personnalisée".
- peut attribuer `IPv6` des adresses aux Pods et aux services, voirthe section called "IPv6".

- vous n'avez pas d'accès Internet sortant, voirthe section called "Clusters privés".

  Pour obtenir la liste complète de toutes les options disponibles et des valeurs par défaut, saisissez la commande suivante.

  ```
  eksctl create nodegroup --help
  ```

  Si les nœuds ne parviennent pas à rejoindre le cluster, reportez-vous the section called "Les nœuds ne parviennent pas à joindre le cluster" au chapitre Dépannage.

  L'exemple qui suit illustre un résultat. Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

  ```
  [#]  created 1 nodegroup(s) in cluster "my-cluster"
  ```

4. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.
5. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

AWS Management Console

Étape 1 : Lancez des nœuds Linux autogérés à l'aide de AWS Management Console

1. Téléchargez la dernière version du AWS CloudFormation modèle.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
   amazon-eks-nodegroup.yaml
   ```

2. Attendez que le statut de votre cluster s'affiche soit `ACTIVE`. Si vous lancez vos nœuds avant que le cluster soit actif, les nœuds ne s'enregistrent pas avec le cluster et vous devrez les relancer.
3. Ouvrez la AWS CloudFormation console.

4. Choisissez Create stack (Créer une pile), puis sélectionnez Avec de nouvelles ressources (standard).
5. Pour Spécifier un modèle, sélectionnez Upload a template file (Télécharger un fichier de modèle), puis sélectionnez Choose file (Choisir un fichier).
6. Sélectionnez le fichier `amazon-eks-nodegroup.yaml` que vous avez téléchargé.
7. Sélectionnez Suivant.
8. Dans la page Specify stack details (Spécifier les détails de la pile), saisissez les paramètres suivants, puis choisissez Next (Suivant) :
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *`my-cluster-nodes`*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - ClusterName: Entrez le nom que vous avez utilisé lors de la création de votre cluster Amazon EKS. Ce nom doit être identique au nom du cluster, sinon vos nœuds ne pourront pas rejoindre le cluster.
   - ClusterControlPlaneSecurityGroup: Choisissez la SecurityGroupsvaleur dans la AWS CloudFormation sortie que vous avez générée lors de la création de votre VPC.

     Les étapes suivantes montrent une opération permettant de récupérer le groupe applicable.

     a. Ouvrez la console Amazon EKS.

     b. Choisissez le nom du cluster.

     c. Choisissez l'onglet Networking (Mise en réseau).

     d. Utilisez la valeur Groupes de sécurité supplémentaires comme référence lorsque vous effectuez une sélection ClusterControlPlaneSecurityGroupdans la liste déroulante.
   - NodeGroupName: Entrez le nom de votre groupe de nœuds. Ce nom pourra être utilisé ultérieurement pour identifier le groupe de nœuds Auto Scaling créé pour vos nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.
   - NodeAutoScalingGroupMinSize: Entrez le nombre minimum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
   - NodeAutoScalingGroupDesiredCapacity: Entrez le nombre de nœuds que vous souhaitez atteindre lors de la création de votre pile.

- NodeAutoScalingGroupMaxSize: Entrez le nombre maximum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeInstanceType: Choisissez un type d'instance pour vos nœuds. Pour de plus amples informations, veuillez consulter the section called "Types d' EC2 instances Amazon".
- NodeImageIdSSMParam: prérempli avec le paramètre Amazon EC2 Systems Manager d'une AMI récemment optimisée pour Amazon EKS pour une version variable de Kubernetes. Pour utiliser une autre version mineure de Kubernetes prise en charge par Amazon EKS, `1.XX` remplacez-la par un autre fichier eks/latest/userguide/kubernetes -versions.html [version prise en charge, type="documentation "]. Nous vous recommandons de spécifier la même version de Kubernetes que celle de votre cluster.

  Vous pouvez également le remplacer `amazon-linux-2` par un autre type d'AMI. Pour de plus amples informations, veuillez consulter the section called "Recevez les dernières nouvelles IDs".

  > **Note**
  >
  > Le nœud Amazon EKS AMIs est basé sur Amazon Linux. Vous pouvez suivre les évènements de sécurité et de confidentialité pour Amazon Linux 2 via le centre de sécurité Amazon Linux ou souscrire au flux RSS associé. Les événements de sécurité et de confidentialité incluent une présentation du problème, les packages concernés et la manière de mettre à jour vos instances pour résoudre le problème.

- NodeImageId: (Facultatif) Si vous utilisez votre propre AMI personnalisée (au lieu d'une AMI optimisée pour Amazon EKS), entrez un ID d'AMI de nœud pour votre AWS région. Si vous spécifiez une valeur ici, elle remplace toutes les valeurs du NodeImageIdSSMParamchamp.
- NodeVolumeSize: Spécifiez une taille de volume racine pour vos nœuds, en GiB.
- NodeVolumeType: Spécifiez un type de volume racine pour vos nœuds.
- KeyName: Entrez le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

> **Note**
>
> Si vous ne fournissez pas de paire de clés ici, la création de la AWS CloudFormation pile échoue.

- BootstrapArguments: Spécifiez tous les arguments facultatifs à transmettre au script bootstrap du nœud, tels que des `kubelet` arguments supplémentaires. Pour de plus amples informations, veuillez consulter Utilisation du script d'amorçage sur GitHub.

  Pour déployer un groupe de nœuds qui :

  - peut attribuer un nombre d'adresses IP nettement supérieur aux pods par rapport à la configuration par défaut, voirthe section called "Augmenter le nombre d'adresses IP".
  - peut attribuer des `IPv4` adresses aux pods à partir d'un bloc CIDR différent de celui de l'instance, voirthe section called "Mise en réseau personnalisée".
  - peut attribuer `IPv6` des adresses aux Pods et aux services, voirthe section called "IPv6".
  - vous n'avez pas d'accès Internet sortant, voirthe section called "Clusters privés".
- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Vous pouvez désactiver IMDSv1. Pour empêcher l'utilisation des futurs nœuds et pods du groupe de nœuds MDSv1, définissez Disable IMDSv1 sur true. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance. Pour plus d'informations sur la façon d'en restreindre l'accès sur vos nœuds, consultez Restreindre l'accès au profil d'instance affecté au composant master.
- VpcId: Entrez l'ID du VPC que vous avez créé.
- Sous-réseaux : choisissez les sous-réseaux que vous avez créés pour votre VPC. Si vous avez créé votre VPC en suivant les étapes décrites dans Créer un Amazon VPC pour votre cluster Amazon EKS, spécifiez uniquement les sous-réseaux privés du VPC sur lesquels vos nœuds doivent se lancer. Vous pouvez consulter les sous-réseaux privés en ouvrant le lien de chaque sous-réseau depuis l'onglet Networking (Mise en réseau) de votre cluster.

> **Important**
>
> - Si certains sous-réseaux sont des sous-réseaux publics, leur paramètre d'attribution automatique d'adresse IP publique doit être activé. Si le paramètre n'est pas activé pour le sous-réseau public, aucun nœud que vous déployez sur ce sous-réseau public ne se verra attribuer d'adresse IP publique et ne pourra pas communiquer

avec le cluster ou d'autres AWS services. Si le sous-réseau a été déployé avant le 26 mars 2020 en utilisant l'un des modèles de AWS CloudFormation VPC Amazon EKS ou `eksctl` en utilisant, l'attribution automatique d'adresses IP publiques est désactivée pour les sous-réseaux publics. Pour plus d'informations sur la façon d'activer l'attribution d'adresses IP publiques pour un sous-réseau, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau. Si le nœud est déployé sur un sous-réseau privé, il est capable de communiquer avec le cluster et d'autres AWS services via une passerelle NAT.

- Si les sous-réseaux n'ont pas accès à Internet, assurez-vous de connaître les considérations et les étapes supplémentaires décrites dans Déployer des clusters privés avec un accès Internet limité.
- Si vous sélectionnez AWS les sous-réseaux Outposts, Wavelength ou Local Zone, les sous-réseaux ne doivent pas avoir été transmis lors de la création du cluster.

9. Sélectionnez les choix que vous souhaitez sur la page Configure stack options (Configurer les options de la pile), puis choisissez Next (Suivant).
10. Cochez la case à gauche de J'accuse réception AWS CloudFormation susceptible de créer des ressources IAM. , puis choisissez Create stack.
11. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
12. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin lors de la configuration de vos nœuds pour Amazon EKS.

Étape 2 : Permettre aux nœuds de rejoindre votre cluster

> **Note**
>
> Si vous avez lancé des nœuds dans un VPC privé sans accès Internet sortant, assurez-vous d'activer les nœuds pour rejoindre votre cluster à partir du VPC.

1. Vérifiez si vous avez déjà appliqué le `ConfigMap aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si vous voyez un `ConfigMap aws-auth`, mettez-le à jour si nécessaire.

a. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Ajoutez une nouvelle entrée `mapRoles` si nécessaire. Définissez la `rolearn` valeur sur la NodeInstanceRolevaleur que vous avez enregistrée lors de la procédure précédente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. Enregistrez le fichier et quittez votre éditeur de texte.

3. Si vous avez reçu un message d'erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le stock `ConfigMap`.

a. Téléchargez la mappe de configuration.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. Dans le `aws-auth-cm.yaml` fichier, définissez la `rolearn` valeur sur celle NodeInstanceRoleque vous avez enregistrée lors de la procédure précédente. Pour ce faire, utilisez un éditeur de texte ou remplacez *`my-node-instance-role`* et exécutez la commande suivante :

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

Saisissez `Ctrl+C` pour revenir à une invite de shell.

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

Si les nœuds ne parviennent pas à rejoindre le cluster, reportez-vous the section called "Les nœuds ne parviennent pas à joindre le cluster" au chapitre Dépannage.

5. (Nœuds GPU uniquement) Si vous avez choisi un type d'instance GPU et l'AMI accélérée optimisée pour Amazon EKS, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes en tant que tel DaemonSet sur votre cluster. *`vX.X.X`*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Étape 3 : actions supplémentaires

1. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.
2. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy (si vous avez `IPv4` un cluster) ou celle (que vous avez créée vous-même si vous avez un `IPv6` cluster) est attachée *`AmazonEKS_CNI_IPv6_Policy`* au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
3. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Créez des nœuds Bottlerocket autogérés

> **Note**
>
> Les groupes de nœuds gérés peuvent offrir certains avantages pour votre cas d'utilisation. Pour de plus amples informations, veuillez consulter the section called "Groupes de nœuds gérés".

Cette rubrique explique comment lancer des groupes Auto Scaling de nœuds Bottlerocket enregistrés auprès de votre cluster Amazon EKS. Bottlerocket est un système d'exploitation open source basé sur Linux AWS que vous pouvez utiliser pour exécuter des conteneurs sur des machines virtuelles ou des hôtes bare metal. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes. Pour plus d'informations sur Bottlerocket, consultez la section Utilisation d'une AMI Bottlerocket avec Amazon EKS activé GitHub et la prise en charge des AMI personnalisées dans la documentation. `eksctl`

Pour plus d'informations sur les mises à niveau sur place, consultez Bottlerocket Update Operator on. GitHub

> **Important**
>
> - Les nœuds Amazon EKS sont EC2 des instances Amazon standard, et ils vous sont facturés sur la base des prix habituels des EC2 instances Amazon. Pour plus d'informations, consultez les EC2 tarifs Amazon.
> - Vous pouvez lancer des nœuds Bottlerocket dans des clusters étendus Amazon EKS sur AWS Outposts, mais vous ne pouvez pas les lancer dans des clusters locaux sur Outposts. AWS Pour de plus amples informations, veuillez consulter Amazon EKS sur AWS Outposts.
> - Vous pouvez effectuer un déploiement sur EC2 des instances Amazon avec `x86` ou des processeurs Arm. Toutefois, vous ne pouvez pas effectuer de déploiement sur des instances dotées de puces Inferentia.

- Bottlerocket est compatible avec. AWS CloudFormation Cependant, aucun CloudFormation modèle officiel ne peut être copié pour déployer des nœuds Bottlerocket pour Amazon EKS.
- Les images Bottlerocket ne sont pas fournies avec un serveur SSH ou un shell. Vous pouvez utiliser des méthodes out-of-band d'accès pour autoriser SSH à activer le conteneur d'administration et pour passer certaines étapes de configuration d'amorçage avec les données utilisateur. Pour plus d'informations, consultez ces sections dans le README.md de bottlerocket sur GitHub :
  - Exploration
  - Conteneur d'administration
  - Paramètres Kubernetes

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour obtenir des instructions sur l'installation ou la mise à niveau`eksctl`, consultez la section Installation de la `eksctl` documentation. Remarque : Cette procédure ne fonctionne que pour les clusters créés avec. `eksctl`

1. Copiez les contenus suivants sur votre appareil. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacer *`ng-bottlerocket`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Pour effectuer un déploiement sur des instances Arm, remplacez *`m5.large`* par un type d'instance Arm. Remplacez-le *`my-ec2-keypair-name`* par le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon. Remplacez tous les *`example values`*

restants par vos propres valeurs. Une fois que vous avez effectué les remplacements, exécutez la commande modifiée pour créer le `bottlerocket.yaml` fichier.

Si vous spécifiez un type d' EC2 instance Arm Amazon, consultez les points à prendre en compte dans Arm Amazon Linux optimisé pour Amazon EKS AMIs avant le déploiement. Pour obtenir des instructions sur le déploiement à l'aide d'une AMI personnalisée, consultez les sections Building Bottlerocket on et GitHub Custom AMI support dans la documentation. `eksctl` Pour déployer un groupe de nœuds gérés, déployez une AMI personnalisée à l'aide d'un modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Modèles de lancement".

> **Important**
>
> Pour déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, AWS Wavelength ou AWS Local Zone, ne transmettez pas les sous-réseaux Outposts AWS , AWS Wavelength ou AWS Local Zone lorsque vous créez le cluster. Vous devez spécifier les sous-réseaux dans l'exemple suivant. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier de configuration et Schéma du fichier de configuration dans la documentation `eksctl`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
```

```
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

2. Deployez vos nœuds avec la commande suivante :

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

L'exemple qui suit illustre un résultat.

Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facultatif) Créez un volume persistant Kubernetes sur un nœud Bottlerocket à l'aide du Plugin CSI Amazon EBS. Le pilote Amazon EBS par défaut repose sur des outils de système de fichiers qui ne sont pas inclus dans Bottlerocket. Pour obtenir plus d'informations sur la création d'une classe de stockage à l'aide du pilote, consultez the section called “Amazon EBS”.
4. (Facultatif) Par défaut, `kube-proxy` définit le paramètre de noyau `nf_conntrack_max` sur une valeur par défaut qui peut différer de ce que Bottlerocket a initialement défini au démarrage. Pour conserver le paramètre par défaut de Bottlerocket, modifiez la `kube-proxy` configuration à l'aide de la commande suivante.

```
kubectl edit -n kube-system daemonset kube-proxy
```

Ajoutez `--conntrack-max-per-core` et `--conntrack-min` aux arguments `kube-proxy` présentés dans l'exemple suivant. Un paramètre de `0` signifie aucun changement.

```
      containers:
      - command:
        - kube-proxy
        - --v=2
```

```
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (Facultatif) Déployer un exemple d'application pour tester vos nœuds Bottlerocket.
6. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Créez des nœuds Microsoft Windows autogérés

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Windows qui s'enregistrent auprès de votre cluster Amazon EKS. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes.

> **Important**
>
> - Les nœuds Amazon EKS sont EC2 des instances Amazon standard, et ils vous sont facturés sur la base des prix habituels des EC2 instances Amazon. Pour plus d'informations, consultez les EC2 tarifs Amazon.
> - Vous pouvez lancer des nœuds Windows dans des clusters étendus Amazon EKS sur AWS Outposts, mais vous ne pouvez pas les lancer dans des clusters locaux sur AWS Outposts. Pour de plus amples informations, veuillez consulter Amazon EKS sur AWS Outposts.

Activer la prise en charge de Windows pour votre cluster avec. Nous vous recommandons de passer en revue les considérations importantes avant de lancer un groupe de nœuds Windows. Pour de plus amples informations, veuillez consulter the section called “Activer le support Windows”.

Vous pouvez lancer des nœuds Windows autogérés avec l'une des options suivantes :

- the section called “`eksctl`”

- the section called "AWS Management Console"

eksctl

Lancez des nœuds Windows autogérés à l'aide de **eksctl**

Cette procédure exige que vous installiez `eksctl` et que votre version `eksctl` soit au moins la version `0.210.0`. Vous pouvez vérifier votre version à l'aide de la commande suivante.

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

> **Note**
>
> Cette procédure fonctionne uniquement pour les clusters créés avec `eksctl`.

1. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy (si vous avez `IPv4` un cluster) ou celle (que vous avez créée vous-même si vous avez un `IPv6` cluster) est attachée *`AmazonEKS_CNI_IPv6_Policy`* au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
2. Cette procédure part du principe que vous disposez d'un cluster existant. Si vous ne disposez pas encore d'un cluster Amazon EKS et d'un groupe de nœuds Amazon Linux auxquels ajouter un groupe de nœuds Windows, nous vous recommandons de suivrethe section called "Créer un cluster (eksctl)". Ce guide fournit une présentation détaillée de la création d'un cluster Amazon EKS avec des nœuds Amazon Linux.

   Créez votre groupe de nœuds avec la commande suivante. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacer *`ng-windows`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une

lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Vous pouvez remplacer *2019* par 2022 pour utiliser Windows Server 2022. Remplacez le reste des *example values* par vos propres valeurs.

> **Important**
>
> Pour déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, AWS Wavelength ou AWS Local Zone, ne transmettez pas les sous-réseaux Outposts AWS , Wavelength ou Local Zone lorsque vous créez le cluster. Créez le groupe de nœuds avec un fichier de configuration, en spécifiant les AWS sous-réseaux Outposts, Wavelength ou Local Zone. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier config et Schéma du fichier Config dans la documentation `eksctl`.

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

> **Note**
>
> - Si les nœuds ne parviennent pas à rejoindre le cluster, reportez-vous à the section called "Les nœuds ne parviennent pas à joindre le cluster" dans le guide de dépannage.
> - Pour voir les options disponibles pour les commandes `eksctl`, saisissez la commande suivante.
>
> ```
> eksctl command -help
> ```

L'exemple qui suit illustre un résultat. Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Windows.
4. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

AWS Management Console

Prérequis

- Un cluster Amazon EKS existant et un groupe de nœuds Linux. Si vous ne disposez pas de ces ressources, nous vous recommandons de les créer à l'aide de l'un de nos guides dansMise en route. Ces guides décrivent comment créer un cluster Amazon EKS avec des nœuds Linux.
- Un VPC et un groupe de sécurité existants qui remplissent les conditions requises pour un cluster Amazon EKS. Pour plus d'informations, consultez the section called "Exigences requises pour le VPC et les sous-réseaux" et the section called "Exigences de groupe de sécurité". Les guides fournis permettent de Mise en route créer un VPC qui répond aux exigences. Vous pouvez également suivre Create an Amazon VPC for your Amazon EKS cluster pour en créer un manuellement.
- Un cluster Amazon EKS existant qui utilise un VPC et un groupe de sécurité qui remplit les conditions requises pour un cluster Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster ". Si vous avez des sous-réseaux dans la AWS région où AWS Outposts, AWS Wavelength ou Local AWS Zones sont activés, ces sous-réseaux ne doivent pas avoir été transmis lorsque vous avez créé le cluster.

Étape 1 : Lancez des nœuds Windows autogérés à l'aide du AWS Management Console

1. Attendez que le statut de votre cluster s'affiche soit `ACTIVE`. Si vous lancez vos nœuds avant que le cluster soit actif, les nœuds ne s'enregistrent pas avec le cluster et vous devez les relancer.

2. Ouvrez la console AWS CloudFormation .
3. Sélectionnez Créer la pile.
4. Dans Spécifier le modèle, sélectionnez URL Amazon S3.
5. Copiez l'URL suivante et collez-la dans l'URL Amazon S3.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Sélectionnez Next (Suivant) deux fois.
7. Sur la page Création rapide d'une pile, saisissez les paramètres suivants en conséquence :
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler `my-cluster-nodes`.
   - ClusterName: Entrez le nom que vous avez utilisé lors de la création de votre cluster Amazon EKS.

     > **Important**
     > Ce nom doit correspondre exactement au nom que vous avez utilisé à l'étape 1 : Création de votre cluster Amazon EKS. Dans le cas contraire, vos nœuds ne pourront pas rejoindre le cluster.

   - ClusterControlPlaneSecurityGroup: Choisissez le groupe de sécurité dans la AWS CloudFormation sortie que vous avez générée lors de la création de votre VPC. Les étapes suivantes montrent une méthode permettant de récupérer le groupe applicable.
     a. Ouvrez la console Amazon EKS.
     b. Choisissez le nom du cluster.
     c. Choisissez l'onglet Networking (Mise en réseau).
     d. Utilisez la valeur Groupes de sécurité supplémentaires comme référence lorsque vous effectuez une sélection ClusterControlPlaneSecurityGroupdans la liste déroulante.
   - NodeGroupName: Entrez le nom de votre groupe de nœuds. Ce nom pourra être utilisé ultérieurement pour identifier le groupe de nœuds Auto Scaling créé pour vos nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

- NodeAutoScalingGroupMinSize: Entrez le nombre minimum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeAutoScalingGroupDesiredCapacity: Entrez le nombre de nœuds que vous souhaitez atteindre lors de la création de votre pile.
- NodeAutoScalingGroupMaxSize: Entrez le nombre maximum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeInstanceType: Choisissez un type d'instance pour vos nœuds. Pour de plus amples informations, veuillez consulter the section called "Types d' EC2 instances Amazon".

> **Note**
>
> Les types d'instances pris en charge pour la dernière version du plugin Amazon VPC CNI pour Kubernetes sont répertoriés dans vpc_ip_resource_limit.go on. GitHub Vous devrez peut-être mettre à jour votre version de CNI pour utiliser les derniers types d'instances pris en charge. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC".

- NodeImageIdSSMParam: prérempli avec le paramètre Amazon EC2 Systems Manager de l'ID AMI Windows Core optimisé pour Amazon EKS actuellement recommandé. Pour utiliser la version complète de Windows, remplacez `Core` par `Full`.
- NodeImageId: (Facultatif) Si vous utilisez votre propre AMI personnalisée (au lieu d'une AMI optimisée pour Amazon EKS), entrez un ID d'AMI de nœud pour votre AWS région. Si vous spécifiez une valeur pour ce champ, elle remplace toutes les valeurs du NodeImageIdSSMParamchamp.
- NodeVolumeSize: Spécifiez une taille de volume racine pour vos nœuds, en GiB.
- KeyName: Entrez le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

> **Note**
>
> Si vous ne fournissez pas de paire de clés ici, la AWS CloudFormation pile ne sera pas créée.

- BootstrapArguments: Spécifiez tous les arguments facultatifs à transmettre au script bootstrap du nœud, tels que des `kubelet` arguments supplémentaires en utilisant-`KubeletExtraArgs`.
- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Vous pouvez le désactiver IMDSv1. Pour empêcher l'utilisation des futurs nœuds et pods du groupe de nœuds MDSv1, définissez Disable IMDSv1 sur true. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance.
- VpcId: Sélectionnez l'ID du VPC que vous avez créé.
- NodeSecurityGroups: Sélectionnez le groupe de sécurité créé pour votre groupe de nœuds Linux lorsque vous avez créé votre VPC. Si vos nœuds Linux sont associés à plusieurs groupes de sécurité, spécifiez-les tous. C'est le cas, par exemple, si le groupe de nœuds Linux a été créé avec `eksctl`.
- Sous-réseaux : choisissez les sous-réseaux que vous avez créés. Si vous avez créé votre VPC en suivant les étapes décrites dans Créer un Amazon VPC pour votre cluster Amazon EKS, spécifiez uniquement les sous-réseaux privés du VPC dans lesquels vos nœuds doivent être lancés.

> **Important**
>
> - Si certains sous-réseaux sont des sous-réseaux publics, leur paramètre d'attribution automatique d'adresse IP publique doit être activé. Si le paramètre n'est pas activé pour le sous-réseau public, aucun nœud que vous déployez sur ce sous-réseau public ne se verra attribuer d'adresse IP publique et ne pourra pas communiquer avec le cluster ou d'autres AWS services. Si le sous-réseau a été déployé avant le 26 mars 2020 en utilisant l'un des modèles de AWS CloudFormation VPC Amazon EKS ou `eksctl` en utilisant, l'attribution automatique d'adresses IP publiques est désactivée pour les sous-réseaux publics. Pour plus d'informations sur la façon d'activer l'attribution d'adresses IP publiques pour un sous-réseau, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau. Si le nœud est déployé sur un sous-réseau privé, il est capable de communiquer avec le cluster et d'autres AWS services via une passerelle NAT.
> - Si les sous-réseaux n'ont pas accès à Internet, assurez-vous de connaître les considérations et les étapes supplémentaires décrites dans Déployer des clusters privés avec un accès Internet limité.

- Si vous sélectionnez AWS les sous-réseaux Outposts, Wavelength ou Local Zone, les sous-réseaux ne doivent pas avoir été transmis lors de la création du cluster.

8. Confirmez que la pile peut créer des ressources IAM, puis choisissez Create stack (Créer une pile).
9. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
10. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin lors de la configuration de vos nœuds Windows Amazon EKS.

Étape 2 : Permettre aux nœuds de rejoindre votre cluster

1. Vérifiez si vous avez déjà appliqué le `ConfigMap` `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si vous voyez un `ConfigMap` `aws-auth`, mettez-le à jour si nécessaire.
   a. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez de nouvelles entrées `mapRoles` selon vos besoins. Définissez les `rolearn` valeurs selon les NodeInstanceRolevaleurs que vous avez enregistrées dans les procédures précédentes.

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
```

```
[...]
```

c. Enregistrez le fichier et quittez votre éditeur de texte.

3. Si vous avez reçu un message d'erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le stock `ConfigMap`.

   a. Téléchargez la mappe de configuration.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
   aws-auth-cm-windows.yaml
   ```

   b. Dans le `aws-auth-cm-windows.yaml` fichier, définissez les `rolearn` valeurs en fonction des NodeInstanceRolevaleurs applicables que vous avez enregistrées dans les procédures précédentes. Pour ce faire, utilisez un éditeur de texte ou remplacez *`example values`* et exécutez la commande suivante :

   ```
   sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-node-
   linux-instance-role|' \
       -e 's|<ARN of windows instance role (not instance profile)>|my-node-windows-
   instance-role|' aws-auth-cm-windows.yaml
   ```

   > **Important**
   >
   > - Ne modifiez aucune autre ligne de ce fichier.
   > - N'utilisez pas le même rôle IAM pour les nœuds Windows et Linux.

   c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

   ```
   kubectl apply -f aws-auth-cm-windows.yaml
   ```

4. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

   ```
   kubectl get nodes --watch
   ```

   Saisissez `Ctrl+C` pour revenir à une invite de shell.

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

Si les nœuds ne parviennent pas à rejoindre le cluster, reportez-vous the section called "Les nœuds ne parviennent pas à joindre le cluster" au chapitre Dépannage.

Étape 3 : actions supplémentaires

1. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Windows.
2. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy (si vous avez `IPv4` un cluster) ou celle (que vous avez créée vous-même si vous avez un `IPv6` cluster) est attachée *`AmazonEKS_CNI_IPv6_Policy`* au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
3. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Créez des nœuds Ubuntu Linux autogérés

> **Note**
>
> Les groupes de nœuds gérés peuvent offrir certains avantages pour votre cas d'utilisation. Pour de plus amples informations, veuillez consulter the section called "Groupes de nœuds gérés".

Cette rubrique explique comment lancer des groupes Auto Scaling d'Ubuntu sur des nœuds Amazon Elastic Kubernetes Service (EKS) ou Ubuntu Pro sur des nœuds Amazon Elastic Kubernetes Service (EKS) enregistrés auprès de votre cluster Amazon EKS. Ubuntu et Ubuntu Pro pour EKS sont basés sur le LTS officiel Ubuntu Minimal, incluent le AWS noyau personnalisé développé conjointement avec AWS EKS et spécialement conçu pour EKS. Ubuntu Pro ajoute une couverture de sécurité supplémentaire en prenant en charge les périodes de support étendues d'EKS, le LivePatch du noyau, la conformité FIPS et la possibilité d'exécuter un nombre illimité de conteneurs Pro.

Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications conteneurisées. Pour plus d'informations, consultez la documentation relative à la prise en charge d'Ubuntu sur Ubuntu AWS et des AMI personnalisées dans la `eksctl` documentation.

> **Important**
>
> - Les nœuds Amazon EKS sont EC2 des instances Amazon standard, et ils vous sont facturés sur la base des prix habituels des EC2 instances Amazon. Pour plus d'informations, consultez les EC2 tarifs Amazon.
> - Vous pouvez lancer des nœuds Ubuntu dans des clusters étendus Amazon EKS sur AWS Outposts, mais vous ne pouvez pas les lancer dans des clusters locaux sur AWS Outposts. Pour de plus amples informations, veuillez consulter Amazon EKS sur AWS Outposts.
> - Vous pouvez effectuer un déploiement sur EC2 des instances Amazon avec `x86` ou des processeurs Arm. Cependant, les instances dotées de puces Inferentia peuvent avoir besoin d'installer d'abord le SDK Neuron.

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour obtenir des instructions sur l'installation ou la mise à niveau`eksctl`, consultez la section Installation de la `eksctl` documentation. Remarque : Cette procédure ne fonctionne que pour les clusters créés avec. `eksctl`

1. Copiez les contenus suivants sur votre appareil. Remplacez `my-cluster` par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphabétique et ne doit pas dépasser 100 caractères. Remplacer `ng-ubuntu` avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Pour effectuer un déploiement sur des instances Arm, remplacez `m5.large` par un type d'instance Arm. Remplacez-le `my-ec2-keypair-name` par le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon. Remplacez tous les *`example values`* restants par vos propres valeurs. Une fois que vous avez effectué les remplacements, exécutez la commande modifiée pour créer le `ubuntu.yaml` fichier.

> **Important**
>
> Pour déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, AWS Wavelength ou AWS Local Zone, ne transmettez pas les sous-réseaux Outposts AWS , AWS Wavelength ou AWS Local Zone lorsque vous créez le cluster. Vous devez spécifier les sous-réseaux dans l'exemple suivant. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier de configuration et Schéma du fichier de configuration dans la documentation `eksctl`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
```

```
metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu2204
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

Pour créer un groupe de nœuds Ubuntu Pro, modifiez simplement la `amiFamily` valeur en`UbuntuPro2204`.

2. Deployez vos nœuds avec la commande suivante :

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

L'exemple qui suit illustre un résultat.

Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facultatif) Déployez un exemple d'application pour tester vos nœuds Ubuntu.
4. Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :

- Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
- Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Mettez à jour les nœuds autogérés pour votre cluster

Quand une nouvelle AMI optimisée pour Amazon EKS est publiée, pensez à remplacer les nœuds autogérés au sein de votre groupe par cette nouvelle AMI. De même, si vous avez mis à jour la version Kubernetes pour votre cluster Amazon EKS, mettez à jour les nœuds afin qu'ils utilisent la même version de Kubernetes.

> **Important**
>
> Cette rubrique couvre les mises à jour des nœuds pour les groupes de nœuds autogérés. Si vous utilisez des groupes de nœuds gérés, consultezthe section called "Mettre à jour".

Deux méthodes de base permettent de mettre à jour les groupes de nœuds autogérés dans vos clusters afin d'utiliser une nouvelle AMI :

Migrer les applications vers un nouveau groupe de nœuds

Créez un nouveau groupe de nœuds et migrez vos pods vers ce groupe. La migration vers un nouveau groupe de nœuds est plus simple que la simple mise à jour de l'ID de l'AMI dans une pile existante AWS CloudFormation . Cela est dû au fait que le processus de migration altère l'ancien groupe de nœuds `NoSchedule` et épuise les nœuds une fois qu'une nouvelle pile est prête à accepter la charge de travail du pod existant.

Mettre à jour une pile de AWS CloudFormation nœuds

Mettez à jour la AWS CloudFormation pile d'un groupe de nœuds existant afin d'utiliser la nouvelle AMI. Cette méthode n'est pas prise en charge pour les groupes de nœuds créés avec`eksctl`.

Migrer les applications vers un nouveau groupe de nœuds

Cette rubrique vous explique comment créer un groupe de nœuds, migrer avec élégance vos applications existantes vers ce nouveau groupe, puis supprimer l'ancien groupe de nœuds de votre cluster. Vous pouvez migrer vers un nouveau groupe de nœuds à l'aide de`eksctl` ou de AWS Management Console.

- the section called "`eksctl`"
- the section called "AWS Management Console et AWS CLI"

`eksctl`

Migrez vos applications vers un nouveau groupe de nœuds avec **`eksctl`**

Pour plus d'informations sur l'utilisation d'eksctl pour la migration, consultez la section Groupes de nœuds non gérés dans la documentation. `eksctl`

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

> **Note**
>
> Cette procédure fonctionne uniquement pour les clusters et les groupes de nœuds créés avec `eksctl`.

1. Récupérez le nom de vos groupes de nœuds existants, en remplaçant *`my-cluster`* par le nom de votre cluster.

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   L'exemple qui suit illustre un résultat.

```
CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
 DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
default      standard-nodes   2019-05-01T22:26:58Z  1             4            3
               t3.medium         ami-05a71d034119ffc12
```

2. Lancez un nouveau groupe de nœuds avec `eksctl` à l'aide de la commande suivante. Dans la commande, remplacez chaque *`example value`* par vos propres valeurs. Le numéro de version ne peut pas être postérieur à la version Kubernetes de votre plan de contrôle. De plus, il ne peut pas y avoir plus de deux versions mineures antérieures à la version de Kubernetes pour votre plan de contrôle. Nous vous recommandons d'utiliser la même version que votre plan de contrôle.

   Nous recommandons de bloquer l'accès du Pod à l'IMDS si les conditions suivantes sont réunies :

   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes afin que les pods ne disposent que des autorisations minimales dont ils ont besoin.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

     Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

     Pour bloquer l'accès du Pod à l'IMDS, ajoutez l'`--disable-pod-imds`option à la commande suivante.

     > **Note**
     >
     > Pour plus d'indicateurs disponibles et leurs descriptions, consultez https://eksctl.io/.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.33 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. Lorsque la commande précédente se termine, vérifiez que tous vos nœuds ont atteint l'état `Ready` à l'aide de la commande suivante :

```
kubectl get nodes
```

4. Supprimez le groupe de nœuds d'origine avec la commande suivante. Dans la commande, remplacez chaque *example value* par les noms de votre cluster et de votre groupe de nœuds :

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console et AWS CLI

Migrez vos applications vers un nouveau groupe de nœuds à l'aide de la AWS CLI AWS Management Console et

1. Lancez un nouveau groupe de nœuds en suivant les étapes décrites dans Créer des nœuds Amazon Linux autogérés.
2. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
3. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin pour ajouter les nouveaux nœuds Amazon EKS à votre cluster.

> **Note**
>
> Si vous avez attachés des stratégies IAM supplémentaires au rôle IAM de votre ancien groupe de nœuds, attachez ces mêmes stratégies au rôle IAM de votre nouveau groupe de nœuds pour maintenir cette fonctionnalité sur le nouveau groupe. Cela s'applique si vous avez ajouté des autorisations pour le Kubernetes Cluster Autoscaler, par exemple.

4. Mettez à jour les groupes de sécurité pour les deux groupes de nœuds afin qu'ils puissent communiquer entre eux. Pour de plus amples informations, veuillez consulter the section called "Exigences de groupe de sécurité".
   a. Enregistrez le groupe de sécurité IDs pour les deux groupes de nœuds. Ceci est affiché sous forme de NodeSecurityGroupvaleur dans les sorties de la AWS CloudFormation pile.

      Vous pouvez utiliser les commandes AWS CLI suivantes pour obtenir le groupe IDs de sécurité à partir des noms de pile. Dans ces commandes, `oldNodes` il s'agit du nom de la AWS CloudFormation pile de votre ancienne pile de nœuds et `newNodes` du nom de la pile vers

laquelle vous effectuez la migration. Remplacez chaque *example value* par vos propres valeurs.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
```

b. Ajoutez des règles d'entrée pour chaque groupe de sécurité des nœuds afin d'accepter le trafic de l'autre.

   Les commandes AWS CLI suivantes ajoutent des règles entrantes à chaque groupe de sécurité qui autorisent tout le trafic sur tous les protocoles en provenance de l'autre groupe de sécurité. Cette configuration permet aux pods de chaque groupe de nœuds de communiquer entre eux pendant que vous migrez votre charge de travail vers le nouveau groupe.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. Modifiez la configmap `aws-auth` afin de mapper le nouveau rôle d'instance des nœuds dans RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Ajoutez une nouvelle entrée `mapRoles` pour le nouveau groupe de nœuds.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
```

```
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Remplacez l'*ARN of instance role (not instance profile)*extrait par la NodeInstanceRolevaleur que vous avez enregistrée lors d'une étape précédente. Ensuite, enregistrez et fermez le fichier pour appliquer le configmap mise à jour.

6. Vérifiez le statut de vos nœuds et attendez que vos nouveaux nœuds rejoignent votre cluster et affichent le statut `Ready`.

```
kubectl get nodes --watch
```

7. (Facultatif) Si vous utilisez le Kubernetes Cluster Autoscaler, réduisez le déploiement à zéro (0) répliques afin d'éviter des actions de dimensionnement contradictoires.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Utilisez la commande suivante pour rejeter chacun des nœuds que vous souhaitez supprimer avec `NoSchedule`. Cela permet d'éviter que les nouveaux pods ne soient planifiés ou reprogrammés sur les nœuds que vous remplacez. Pour plus d'informations, consultez Taints and Tolerations dans la documentation de Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

Si vous mettez à niveau vos nœuds vers une nouvelle version de Kubernetes, vous pouvez identifier et altérer tous les nœuds d'une version particulière de Kubernetes (dans ce cas,`1.31`) à l'aide de l'extrait de code suivant. Le numéro de version ne peut pas être postérieur à la version Kubernetes de votre plan de contrôle. Il ne peut pas non plus y avoir plus de deux versions mineures antérieures à la version Kubernetes de votre plan de contrôle. Nous vous recommandons d'utiliser la même version que votre plan de contrôle.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
```

```
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. Déterminez le fournisseur DNS de votre cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

L'exemple qui suit illustre un résultat. Ce cluster utilise CoreDNS pour la résolution DNS, mais votre cluster peut `kube-dns` renvoyer à la place) :

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. Si votre déploiement actuel exécute moins de deux réplicas, dimensionnez le déploiement à deux réplicas. Remplacez *`coredns`* par `kubedns` si la sortie de votre commande a plutôt renvoyé cette valeur.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. Drainez chacun des nœuds que vous souhaitez supprimer de votre cluster à l'aide de la commande suivante :

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

Si vous mettez à niveau vos nœuds vers une nouvelle version de Kubernetes, identifiez et videz tous les nœuds d'une version particulière de Kubernetes (dans ce cas,*`1.31`*) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. Une fois que vos anciens nœuds ont été purgés, révoquez les règles d'entrée du groupe de sécurité que vous avez autorisées précédemment. Supprimez ensuite la AWS CloudFormation pile pour mettre fin aux instances.

> **Note**
>
> Si vous avez associé des politiques IAM supplémentaires à votre ancien rôle IAM de groupe de nœuds, par exemple en ajoutant des autorisations pour le Kubernetes Cluster Autoscaler, détachez-les du rôle avant de supprimer votre stack. AWS CloudFormation

a. Révoquez les règles entrantes que vous aviez créées précédemment pour les groupes de sécurité de vos nœuds. Dans ces commandes, `oldNodes` il s'agit du nom de la AWS CloudFormation pile de votre ancienne pile de nœuds et `newNodes` du nom de la pile vers laquelle vous effectuez la migration.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. Ouvrez la AWS CloudFormation console.

c. Sélectionnez votre ancienne pile de nœuds.

d. Sélectionnez Delete (Supprimer).

e. Dans la boîte de dialogue de confirmation Delete stack (Supprimer la pile), choisissez Delete stack (Supprimer la pile).

13. Modifiez la configmap `aws-auth` afin de supprimer l'ancien rôle d'instance des nœuds dans RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Supprimez l'entrée `mapRoles` de l'ancien groupe de nœuds.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

Enregistrez et fermez le fichier afin d'appliquer la configmap mise à jour.

14. (Facultatif) Si vous utilisez le composant Cluster Autoscaler de Kubernetes, dimensionnez le déploiement à un réplica.

> **Note**
>
> Vous devez également labeliser votre nouveau groupe Auto Scaling de façon appropriée (par exemple `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/my-cluster`) et mettre à jour la commande de votre déploiement Cluster Autoscaler afin qu'elle pointe vers le groupe Auto Scaling nouvellement labelisé. Pour plus d'informations, consultez Cluster Autoscaler activé. AWS

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (Facultatif) Vérifiez que vous utilisez la dernière version du plugin Amazon VPC CNI pour Kubernetes. Vous devrez peut-être mettre à jour votre version de CNI pour utiliser les derniers

types d'instances pris en charge. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC".

16. Si votre cluster utilise la résolution `kube-dns` DNS (voir[migrate-determine-dns-step]), adaptez le `kube-dns` déploiement à une seule réplique.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

Mettre à jour une pile de AWS CloudFormation nœuds

Cette rubrique décrit comment mettre à jour une pile de nœuds AWS CloudFormation autogérés existante avec une nouvelle AMI. Vous pouvez utiliser cette procédure pour mettre à jour vos nœuds vers une nouvelle version de Kubernetes suite à une mise à jour du cluster. Sinon, vous pouvez effectuer une mise à jour vers la dernière AMI optimisée pour Amazon EKS pour une version Kubernetes existante.

> **Important**
>
> Cette rubrique couvre les mises à jour des nœuds pour les groupes de nœuds autogérés. Pour plus d'informations sur l'utilisation de Simplifier le cycle de vie des nœuds avec des groupes de nœuds gérés, consultezthe section called "Mettre à jour".

Le dernier AWS CloudFormation modèle de nœud Amazon EKS par défaut est configuré pour lancer une instance avec la nouvelle AMI dans votre cluster avant de supprimer l'ancienne, une par une. Cette configuration garantit que vous disposez toujours du nombre d'instances actives souhaité par votre groupe Auto Scaling dans votre cluster lors de la mise à jour continue.

> **Note**
>
> Cette méthode n'est pas prise en charge pour les groupes de nœuds créés avec`eksctl`. Si vous avez créé votre cluster ou votre groupe de nœuds avec `eksctl`, consultez the section called "Migration".

1. Déterminez le fournisseur DNS de votre cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

L'exemple qui suit illustre un résultat. Ce cluster utilise CoreDNS pour la résolution DNS, mais il se peut que votre cluster réapparaisse à la place. `kube-dns` Votre sortie peut sembler différente selon la version `kubectl` que vous utilisez.

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

2. Si votre déploiement actuel exécute moins de deux réplicas, dimensionnez le déploiement à deux réplicas. Remplacez *`coredns`* par `kube-dns` si la sortie de votre commande a plutôt renvoyé cette valeur.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

3. (Facultatif) Si vous utilisez le Kubernetes Cluster Autoscaler, réduisez le déploiement à zéro (0) répliques afin d'éviter des actions de dimensionnement contradictoires.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. Déterminez le type d'instance et le nombre d'instances souhaité pour votre groupe de nœuds actuel. Vous saisirez ces valeurs ultérieurement lorsque vous mettrez à jour le AWS CloudFormation modèle du groupe.
   a. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
   b. Dans le panneau de navigation de gauche, sélectionnez Launch Configurations (Configurations du lancement) et notez le type d'instance pour votre configuration actuelle de lancement des nœuds.
   c. Dans le panneau de navigation de gauche, sélectionnez Auto Scaling Groups (Groupes Auto Scaling) et notez le nombre d'instances souhaité pour le groupe Auto Scaling actuel des nœuds.
5. Ouvrez la AWS CloudFormation console.
6. Sélectionnez votre pile de groupe de nœuds, puis choisissez Update (Mettre à jour).
7. Sélectionnez Replace current template (Remplacer le modèle actuel) et sélectionnez Amazon S3 URL (URL Amazon S3).
8. Pour l'URL Amazon S3, collez l'URL suivante dans la zone de texte pour vous assurer que vous utilisez la dernière version du AWS CloudFormation modèle de nœud. Ensuite, sélectionnez Next (Suivant) :

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-
nodegroup.yaml
```

9. Sur la page Specify stack details (Spécifier les détails de la pile), renseignez les paramètres suivants, puis choisissez Next (Suivant) :
   - NodeAutoScalingGroupDesiredCapacity— Entrez le nombre d'instances souhaité que vous avez enregistré lors d'une étape précédente. Ou saisissez le nouveau nombre de nœuds souhaité pour la mise à l'échelle lorsque votre pile sera mise à jour.
   - NodeAutoScalingGroupMaxSize— Entrez le nombre maximum de nœuds auxquels votre groupe Auto Scaling de nœuds peut s'étendre. Cette valeur doit être supérieure d'au moins un nœud à votre capacité souhaitée. Ceci afin de pouvoir effectuer une mise à jour continue de vos nœuds sans réduire votre nombre de nœuds pendant la mise à jour.
   - NodeInstanceType— Choisissez le type d'instance que vous avez enregistré à l'étape précédente. Vous pouvez également choisir un type d'instance différent pour vos nœuds. Avant de choisir un autre type d'instance, consultez Choisir un type d'instance de EC2 nœud Amazon optimal. Chaque type d' EC2 instance Amazon prend en charge un nombre maximum d'interfaces réseau élastiques (interface réseau) et chaque interface réseau prend en charge un nombre maximum d'adresses IP. Étant donné que chaque nœud de travail et chaque pod se voient attribuer sa propre adresse IP, il est important de choisir un type d'instance qui prendra en charge le nombre maximum de pods que vous souhaitez exécuter sur chaque EC2 nœud Amazon. Pour une liste du nombre d'interfaces réseau et d'adresses IP pris en charge par les types d'instances, consultez adresses IP par interface réseau et par type d'instance. Par exemple, le type d'`m5.large`instance prend en charge un maximum de 30 adresses IP pour le nœud de travail et les pods.

     > **Note**
     >
     > Les types d'instances pris en charge pour la dernière version du plugin CNI d'Amazon VPC pour Kubernetes sont indiqués dans vpc_ip_resource_limit.go sur GitHub. Vous devrez peut-être mettre à jour votre plug-in Amazon VPC CNI pour Kubernetes afin d'utiliser les derniers types d'instances pris en charge. Pour de plus amples informations, veuillez consulter the section called “CNI Amazon VPC”.

> **Important**
>
> Certains types d'instances peuvent ne pas être disponibles dans toutes les AWS régions.

- NodeImageIdSSMParam— Le paramètre Amazon EC2 Systems Manager de l'ID AMI vers lequel vous souhaitez effectuer la mise à jour. La valeur suivante utilise la dernière version de l'AMI optimisée Amazon EKS pour Kubernetes. `1.33`

```
/aws/service/eks/optimized-ami/1.33/amazon-linux-2/recommended/image_id
```

  Vous pouvez le *`1.33`* remplacer par un eks/latest/userguide/platform fichier -versions.html [platform-version, type="documentation "] identique. Ou bien, elle doit être antérieure d'une version au maximum à la version Kubernetes exécutée sur votre plan de contrôle. Nous vous recommandons de conserver vos nœuds dans la même version que votre plan de contrôle. Vous pouvez également le remplacer *`amazon-linux-2`* par un autre type d'AMI. Pour de plus amples informations, veuillez consulter the section called "Recevez les dernières nouvelles IDs".

> **Note**
>
> L'utilisation du paramètre Amazon EC2 Systems Manager vous permet de mettre à jour vos nœuds à l'avenir sans avoir à rechercher et à spécifier un ID AMI. Si votre AWS CloudFormation stack utilise cette valeur, toute mise à jour de stack lance toujours la dernière AMI optimisée Amazon EKS recommandée pour la version de Kubernetes que vous avez spécifiée. C'est même le cas même si vous ne modifiez aucune valeur dans le modèle.

- NodeImageId— Pour utiliser votre propre AMI personnalisée, entrez l'ID de l'AMI à utiliser.

> **Important**
>
> Cette valeur remplace toute valeur spécifiée pour NodeImageIdSSMParam. Si vous souhaitez utiliser la NodeImageIdSSMParamvaleur, assurez-vous que la valeur pour NodeImageIdest vide.

- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Cependant, vous pouvez le désactiver IMDSv1. Sélectionnez true si vous ne souhaitez pas que les nœuds ou les pods planifiés dans le groupe de nœuds soient utilisés IMDSv1. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance. Si vous avez implémenté des rôles IAM pour les comptes de service, attribuez les autorisations nécessaires directement à tous les pods nécessitant un accès aux AWS services. Ainsi, aucun pod de votre cluster n'a besoin d'accéder à l'IMDS pour d'autres raisons, telles que la récupération de la région actuelle AWS . Ensuite, vous pouvez également désactiver l'accès IMDSv2 aux pods qui n'utilisent pas le réseau hôte. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

10. (Facultatif) Sur la page Options (Options), étiquetez les ressources de votre pile. Choisissez Suivant.
11. Sur la page Review (Vérification), vérifiez vos informations, acceptez que la pile puisse créer des ressources IAM, puis choisissez Update stack (Mettre à jour la pile).

> **Note**
>
> La mise à jour de chaque nœud du cluster prend plusieurs minutes. Attendez que la mise à jour de tous les nœuds soit terminée avant d'effectuer les étapes suivantes.

12. Si le fournisseur DNS de votre cluster l'est`kube-dns`, adaptez le `kube-dns` déploiement à une seule réplique.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (Facultatif) Si vous utilisez le composant Cluster Autoscaler de Kubernetes, redimensionnez le déploiement à la quantité souhaitée de réplicas.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (Facultatif) Vérifiez que vous utilisez la dernière version du plugin Amazon VPC CNI pour Kubernetes. Vous devrez peut-être mettre à jour votre plug-in Amazon VPC CNI pour Kubernetes afin d'utiliser les derniers types d'instances pris en charge. Pour de plus amples informations, veuillez consulter the section called “CNI Amazon VPC”.

Simplifiez la gestion des calculs avec AWS Fargate

Cette rubrique décrit l'utilisation d'Amazon EKS pour exécuter des pods Kubernetes sur Fargate. AWS Fargate est une technologie qui fournit une capacité de calcul à la demande et de taille appropriée pour les conteneurs. Avec Fargate, vous n'avez pas à approvisionner, configurer ou dimensionner vous-même des groupes de machines virtuelles pour exécuter des conteneurs. Vous n'avez pas non plus besoin de choisir les types de serveurs, de décider quand dimensionner vos groupes de nœuds ou d'optimiser le regroupement de clusters.

Vous pouvez contrôler quels pods démarrent sur Fargate et comment ils fonctionnent avec les profils Fargate. Les profils Fargate sont définis dans le cadre de votre cluster Amazon EKS. Amazon EKS intègre Kubernetes à Fargate à l'aide de contrôleurs conçus à l'aide du modèle extensible en amont fourni par AWS Kubernetes. Ces contrôleurs s'exécutent dans le cadre du plan de contrôle Kubernetes géré par Amazon EKS et sont chargés de planifier les pods Kubernetes natifs sur Fargate. Les contrôleurs Fargate comprennent un nouveau planificateur qui s'exécute parallèlement au planificateur Kubernetes par défaut, en plus de plusieurs contrôleurs d'admission mutants et validants. Lorsque vous démarrez un Pod répondant aux critères d'exécution sur Fargate, les contrôleurs Fargate exécutés dans le cluster reconnaissent, mettent à jour et programment le Pod sur Fargate.

Cette rubrique décrit les différents composants des pods qui s'exécutent sur Fargate et présente des points particuliers à prendre en compte lors de l'utilisation de Fargate avec Amazon EKS.

AWS Considérations relatives à Fargate

Voici quelques éléments à prendre en compte pour l'utilisation de Fargate sur Amazon EKS.

- Chaque pod qui s'exécute sur Fargate possède sa propre limite de calcul. Ils ne partagent pas le noyau sous-jacent, les ressources du processeur, les ressources de mémoire ou l'interface Elastic Network avec un autre Pod.
- Les équilibreurs de charge réseau et les équilibreurs de charge d'application (ALBs) ne peuvent être utilisés avec Fargate qu'avec des cibles IP. Pour plus d'informations, consultez the section called "Créer un équilibreur de charge de réseau" et the section called "Répartition de la charge des applications".
- Les services exposés de Fargate s'exécutent uniquement en mode IP de type cible, et non en mode IP de nœud. La manière recommandée de vérifier la connectivité entre un service s'exécutant sur un nœud géré et un service s'exécutant sur Fargate est de se connecter via le nom du service.

- Les pods doivent correspondre à un profil Fargate au moment où leur exécution est prévue sur Fargate. Les pods qui ne correspondent pas à un profil Fargate peuvent être bloqués en tant que. `Pending` S'il existe un profil Fargate correspondant, vous pouvez supprimer les pods en attente que vous avez créés pour les reprogrammer sur Fargate.
- Les daemonsets ne sont pas pris en charge sur Fargate. Si votre application nécessite un démon, reconfigurez ce démon pour qu'il s'exécute en tant que conteneur annexe dans vos pods.
- Les conteneurs privilégiés ne sont pas pris en charge sur Fargate.
- Les pods exécutés sur Fargate ne peuvent pas le `HostPort` spécifier `HostNetwork` ou dans le manifeste du Pod.
- La limite par défaut `nofile` et `nproc` souple est de 1024 et la limite stricte est de 65535 pour les Fargate Pods.
- GPUs ne sont actuellement pas disponibles sur Fargate.
- Les pods qui s'exécutent sur Fargate ne sont pris en charge que sur les sous-réseaux privés (avec accès par passerelle NAT AWS aux services, mais pas de route directe vers une passerelle Internet). Le VPC de votre cluster doit donc disposer de sous-réseaux privés disponibles. Pour les clusters sans accès Internet sortant, veuillez consulter the section called "Clusters privés".
- Vous pouvez utiliser l'outil Adjust pod resources with Vertical Pod Autoscaler pour définir la taille initiale correcte du processeur et de la mémoire de vos Fargate Pods, puis utiliser les déploiements de pods Scale avec Horizontal Pod Autoscaler pour dimensionner ces pods. Si vous souhaitez que le Vertical Pod Autoscaler redéploie automatiquement les pods vers Fargate avec des combinaisons de processeur et de mémoire plus importantes, réglez le mode du Vertical Pod Autoscaler sur l'un ou l'autre ou pour garantir un fonctionnement correct. `Auto` `Recreate` Pour plus d'informations, consultez la documentation Vertical Pod Autoscaler sur GitHub.
- La résolution DNS et les noms d'hôte DNS doivent être activés pour votre VPC. Pour plus d'informations, consultez Affichage et mise à jour de la prise en charge de DNS pour votre VPC.
- Amazon EKS Fargate defense-in-depth complète les applications Kubernetes en isolant chaque pod au sein d'une machine virtuelle (VM). Cette frontière VM empêche l'accès aux ressources basées sur l'hôte utilisées par d'autres pods en cas de fuite du conteneur, qui est une méthode courante d'attaque des applications conteneurisées et d'accès aux ressources en dehors du conteneur.

  L'utilisation d'Amazon EKS ne modifie pas vos responsabilités dans le cadre du modèle de responsabilité partagée. Vous devez examiner attentivement la configuration des contrôles de sécurité et de gouvernance du cluster. Le moyen le plus sûr d'isoler une application est toujours de l'exécuter dans un cluster séparé.

- Les profils Fargate prennent en charge la spécification de sous-réseaux à partir de blocs CIDR secondaires de VPC. Vous pouvez spécifier un bloc CIDR secondaire. Cela est dû au fait que le nombre d'adresses IP disponibles dans un sous-réseau est limité. Par conséquent, le nombre de pods pouvant être créés dans le cluster est également limité. En utilisant différents sous-réseaux pour les pods, vous pouvez augmenter le nombre d'adresses IP disponibles. Pour plus d'informations, consultez la section Ajout de blocs IPv4 CIDR à un VPC.
- Le service de métadonnées d' EC2 instance Amazon (IMDS) n'est pas disponible pour les pods déployés sur les nœuds Fargate. Si certains de vos pods déployés sur Fargate nécessitent des informations d'identification IAM, attribuez-les à vos pods en utilisant les rôles IAM pour les comptes de service. Si vos pods ont besoin d'accéder à d'autres informations disponibles via IMDS, vous devez coder ces informations en dur dans les spécifications de votre pod. Cela inclut la AWS région ou la zone de disponibilité dans laquelle un pod est déployé.
- Vous ne pouvez pas déployer de Fargate Pods dans des AWS Outposts AWS , Wavelength ou des Zones Locales. AWS
- Amazon EKS doit appliquer régulièrement des correctifs aux Fargate Pods pour garantir leur sécurité. Nous essayons de procéder aux mises à jour de manière à réduire l'impact, mais il arrive parfois que les pods doivent être supprimés s'ils ne sont pas expulsés avec succès. Certaines actions peuvent être prises pour minimiser les perturbations. Pour de plus amples informations, veuillez consulter the section called “Événements relatifs aux correctifs du système d'exploitation”.
- Le pluginAmazon VPC CNI pour Amazon EKS est installé sur des nœuds Fargate. Vous ne pouvez pas utiliser de plug-ins CNI alternatifs pour les clusters Amazon EKS dotés de nœuds Fargate.
- Un Pod exécuté sur Fargate monte automatiquement un système de fichiers Amazon EFS, sans nécessiter d'étapes d'installation manuelle du pilote. Vous ne pouvez pas utiliser le provisionnement dynamique de volumes persistants avec les nœuds Fargate, mais vous pouvez utiliser le provisionnement statique.
- Amazon EKS ne prend pas en charge Fargate Spot.
- Vous ne pouvez pas monter de volumes Amazon EBS sur des Fargate Pods.
- Vous pouvez exécuter le contrôleur Amazon EBS CSI sur des nœuds Fargate, mais le nœud DaemonSet Amazon EBS CSI ne peut fonctionner que sur des instances Amazon. EC2
- Une fois qu'un Job Kubernetes est `Completed` marqué `Failed` ou, les pods créés par le Job continuent normalement d'exister. Ce comportement vous permet de consulter vos journaux et vos résultats, mais avec Fargate, vous devrez payer des frais si vous ne nettoyez pas le Job par la suite.

Pour supprimer automatiquement les pods associés après la fin ou l'échec d'un Job, vous pouvez spécifier une période à l'aide du contrôleur time-to-live (TTL). L'exemple suivant montre comment spécifier `.spec.ttlSecondsAfterFinished` dans votre manifeste Job.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Tableau comparatif de Fargate

Critères	AWS Fargate
Peut être déployé dans les AWS Outposts	Non
Possibilité d'être déployé dans une zone locale AWS	Non
Possibilité d'exécuter des conteneurs qui nécessitent Windows	Non
Possibilité d'exécuter des conteneurs qui nécessitent Linux	Oui
Possibilité d'exécuter des applications nécessitant la puce Inferentia	Non
Possibilité d'exécuter des applications nécessitant un GPU	Non

Critères	AWS Fargate
Possibilité d'exécuter des applications nécessitant des processeurs Arm	Non
Possibilité d'exécuter AWS Bottlerocket	Non
Les pods partagent un environnement d'exécution du noyau avec d'autres pods	Non — Chaque pod possède un noyau dédié
Les pods partagent le processeur, la mémoire, le stockage et les ressources réseau avec d'autres pods.	Non, chaque pod dispose de ressources dédiées et peut être dimensionné indépendamment pour optimiser l'utilisation des ressources.
Les pods peuvent utiliser plus de matériel et de mémoire que ce qui est demandé dans les spécifications des pods	Non, le Pod peut cependant être redéployé à l'aide d'une configuration de vCPU et de mémoire plus importante.
Doit déployer et gérer des EC2 instances Amazon	Non
Doit sécuriser, maintenir et appliquer des correctifs au système d'exploitation des EC2 instances Amazon	Non
Peut fournir des arguments bootstrap lors du déploiement d'un nœud, tels que des arguments kubelet supplémentaires.	Non
Peut attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de l'adresse IP attribuée au nœud.	Non
Possibilité d'accéder au nœud via SSH	Non — Il n'existe aucun système d'exploitation hôte de nœud auquel se connecter par SSH.
Possibilité de déployer votre propre AMI personnalisée sur les nœuds	Non

Critères	AWS Fargate
Possibilité de déployer votre propre CNI personnalisée sur les nœuds	Non
Vous devez mettre à jour l'AMI du nœud par vous-même	Non
Vous devez mettre à jour la version du nœud Kubernetes par vous-même	Non, vous ne gérez pas les nœuds.
Peut utiliser le stockage Amazon EBS avec des pods	Non
Peut utiliser le stockage Amazon EFS avec des pods	Oui
Peut utiliser Amazon FSx pour le stockage de Lustre avec des pods	Non
Possibilité d'utiliser le Network Load Balancer pour les services	Oui, lors de l'utilisation de l'outil Créer un équilibreur de charge réseau
Les pods peuvent s'exécuter dans un sous-rése au public	Non
Possibilité d'attribuer différents groupes de sécurité VPC à des pods individuels	Oui
Peut exécuter Kubernetes DaemonSets	Non
Support `HostPort` et manifeste `HostNetwo rk` intégré au Pod	Non
AWS Disponibilité de la région	Quelques régions prises en charge par Amazon EKS
Peut exécuter des conteneurs sur des hôtes EC2 dédiés Amazon	Non

Critères	AWS Fargate
Tarification	Coût d'une mémoire Fargate individuelle et d'une configuration CPU. Chaque Pod a son propre coût. Pour plus d'informations, consultez Tarification de AWS Fargate.

Commencez à utiliser AWS Fargate pour votre cluster

Cette rubrique explique comment commencer à exécuter des pods sur AWS Fargate avec votre cluster Amazon EKS.

Si vous limitez l'accès au point de terminaison public de votre cluster à l'aide de blocs CIDR, nous vous recommandons d'activer également l'accès au point de terminaison privé. De cette façon, les Fargate Pods peuvent communiquer avec le cluster. Sans l'activation du point de terminaison privé, les blocs CIDR que vous spécifiez pour l'accès public doivent inclure les sources de sortie de votre VPC. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

Prérequis

Un cluster existant. Si vous ne possédez pas encore de cluster Amazon EKS, consultezMise en route.

Étape 1 : Assurez-vous que les nœuds existants peuvent communiquer avec les Fargate Pods

Si vous travaillez avec un nouveau cluster sans nœuds, ou un cluster avec uniquement des groupes de nœuds gérés (voirthe section called "Groupes de nœuds gérés"), vous pouvez passer àthe section called "Étape 2 : Création d'un rôle d'exécution Fargate Pod".

Supposons que vous travaillez avec un cluster existant auquel des nœuds sont déjà associés. Assurez-vous que les pods de ces nœuds peuvent communiquer librement avec les pods qui s'exécutent sur Fargate. Les pods qui s'exécutent sur Fargate sont automatiquement configurés pour utiliser le groupe de sécurité du cluster auquel ils sont associés. Assurez-vous que tous les nœuds existants de votre cluster peuvent envoyer et recevoir du trafic vers et depuis le groupe de sécurité du cluster. Les groupes de nœuds gérés sont automatiquement configurés pour utiliser également

le groupe de sécurité du cluster. Vous n'avez donc pas besoin de les modifier ou de vérifier leur compatibilité (voirthe section called "Groupes de nœuds gérés").

Pour les groupes de nœuds existants créés avec `eksctl` ou avec les AWS CloudFormation modèles gérés par Amazon EKS, vous pouvez ajouter le groupe de sécurité du cluster aux nœuds manuellement. Vous pouvez également modifier le modèle de lancement du groupe Auto Scaling pour le groupe de nœuds, afin d'attacher le groupe de sécurité du cluster aux instances. Pour plus d'informations, consultez la section Modification des groupes de sécurité d'une instance dans le guide de l'utilisateur Amazon VPC.

Vous pouvez rechercher un groupe de sécurité pour votre cluster dans la AWS Management Console section Mise en réseau du cluster. Vous pouvez également le faire à l'aide de la commande AWS CLI suivante. Lorsque vous utilisez cette commande, remplacez `<my-cluster>` par le nom de votre cluster.

```
aws eks describe-cluster --name <my-cluster> --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Étape 2 : Création d'un rôle d'exécution Fargate Pod

Lorsque votre cluster crée des pods sur AWS Fargate, les composants qui s'exécutent sur l'infrastructure Fargate doivent effectuer des appels en votre nom. AWS APIs Le rôle d'exécution Amazon EKS Pod fournit les autorisations IAM nécessaires à cette fin. Pour créer un rôle d'exécution AWS Fargate Pod, consultez. the section called "Rôle IAM d'exécution de pod"

> **Note**
>
> Si vous avez créé votre cluster à `eksctl` l'aide de `--fargate` cette option, celui-ci possède déjà un rôle d'exécution Pod que vous pouvez trouver dans la console IAM avec le modèle`eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`. De même, si vous créez `eksctl` vos profils Fargate`eksctl`, crée votre rôle d'exécution Pod s'il n'en est pas déjà créé un.

Étape 3 : Créez un profil Fargate pour votre cluster

Avant de pouvoir planifier des pods exécutés sur Fargate dans votre cluster, vous devez définir un profil Fargate qui indique quels pods utilisent Fargate lors de leur lancement. Pour de plus amples informations, veuillez consulter the section called "Définir des profils".

> **Note**
>
> Si vous avez créé votre cluster à `eksctl` l'aide de `--fargate` cette option, un profil Fargate est déjà créé pour votre cluster avec des sélecteurs pour tous les pods dans les espaces de noms et. `kube-system` `default` Utilisez la procédure suivante pour créer des profils Fargate pour tout autre espace de noms que vous souhaitez utiliser avec Fargate.

Vous pouvez créer un profil Fargate à l'aide de l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Cette procédure nécessite `eksctl` version `0.210.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

Pour créer un profil Fargate avec **`eksctl`**

Créez votre profil Fargate avec la commande `eksctl` suivante, en remplaçant chaque `<example value>` par vos propres valeurs. Vous devez spécifier un espace de noms. Cependant, `--labels` cette option n'est pas obligatoire.

```
eksctl create fargateprofile \
    --cluster <my-cluster> \
    --name <my-fargate-profile> \
    --namespace <my-kubernetes-namespace> \
    --labels <key=value>
```

Vous pouvez utiliser certains caractères génériques pour les étiquettes `<my-kubernetes-namespace>` et `<key=value>`. Pour de plus amples informations, veuillez consulter the section called "Caractères génériques de profils Fargate".

AWS Management Console

Pour créer un profil Fargate avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster pour lequel vous voulez créer un profil Fargate.
3. Choisissez l'onglet Calcul.
4. Sous Fargate profiles (Profils Fargate), choisissez Add Fargate profile (Ajouter un profil Fargate).
5. Sur la page Configure Fargate profile (Configurer le profil Fargate), procédez comme suit :
   a. Dans Nom, saisissez un nom pour votre profil Fargate. Le nom doit être unique.
   b. Pour le rôle d'exécution du Pod, choisissez le rôle d'exécution du Pod à utiliser avec votre profil Fargate. Seuls les rôles IAM avec le principal de service `eks-fargate-pods.amazonaws.com` sont affichés. Si aucun rôle n'est répertorié, vous devez en créer un. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM d'exécution de pod".
   c. Modifiez les sous-réseaux sélectionnés selon vos besoins.

      > **Note**
      > Seuls les sous-réseaux privés sont pris en charge pour les pods exécutés sur Fargate.

   d. Dans Identifications, vous pouvez éventuellement étiqueter votre profil Fargate. Ces balises ne se propagent pas aux autres ressources associées au profil, telles que les pods.
   e. Choisissez Suivant.
6. Sur la page de sélection Configurer le module, procédez comme suit :
   a. Pour Namespace, entrez un espace de noms correspondant aux pods.
      - Vous pouvez utiliser des espaces de noms spécifiques pour les faire correspondre, tels que `kube-system` ou `default`.
      - Vous pouvez utiliser certains caractères génériques (par exemple, `prod-*`) pour faire correspondre plusieurs espaces de noms (par exemple, `prod-deployment` et `prod-test`). Pour de plus amples informations, veuillez consulter the section called "Caractères génériques de profils Fargate".
   b. (Facultatif) Ajoutez des labels Kubernetes au sélecteur. Ajoutez-les spécifiquement à celui que les pods de l'espace de noms spécifié doivent correspondre.

- Vous pouvez ajouter l'étiquette `infrastructure: fargate` au sélecteur afin que seuls les pods de l'espace de noms spécifié qui portent également l'étiquette `infrastructure: fargate` Kubernetes correspondent au sélecteur.
- Vous pouvez utiliser certains caractères génériques (par exemple, `key?: value?`) pour faire correspondre plusieurs espaces de noms (par exemple, `keya: valuea` et `keyb: valueb`). Pour de plus amples informations, veuillez consulter the section called "Caractères génériques de profils Fargate".

c. Choisissez Suivant.

7. Sur la page Vérifier et créer, vérifiez les informations de votre profil Fargate et choisissez Créer.

Étape 4 : Mettre à jour CoreDNS

Par défaut, CoreDNS est configuré pour s'exécuter sur l' EC2 infrastructure Amazon sur des clusters Amazon EKS. Si vous souhaitez uniquement exécuter vos pods sur Fargate dans votre cluster, procédez comme suit.

> **Note**
>
> Si vous avez créé votre cluster à l'aide de `eksctl` en utilisant l'option `--fargate`, vous pouvez passer directement à the section called "Étapes suivantes".

1. Créez un profil Fargate pour CoreDNS à l'aide de la commande suivante. Remplacez-le `<my-cluster>` par le nom de votre cluster, `<111122223333>` par votre identifiant de compte, `<AmazonEKSFargatePodExecutionRole>` par le nom de votre rôle d'exécution de Pod et `<000000000000000c>` par le nom IDs de vos sous-réseaux privés. `<000000000000000a>` `<000000000000000b>` Si vous n'avez pas de rôle d'exécution de Pod, vous devez d'abord en créer un (voirthe section called "Étape 2 : Création d'un rôle d'exécution Fargate Pod").

   > **Important**
   >
   > L'ARN du rôle ne peut pas inclure de chemin autre que/. Par exemple, si le nom de votre rôle est `development/apps/AmazonEKSFargatePodExecutionRole`, vous devez le remplacer par `AmazonEKSFargatePodExecutionRole` lorsque vous spécifiez l'ARN du rôle. Le format de l'ARN de rôle doit être `arn:aws: iam::<111122223333>:role/<AmazonEKSFargatePodExecutionRole>`.

```
aws eks create-fargate-profile \
    --fargate-profile-name coredns \
    --cluster-name <my-cluster> \
    --pod-execution-role-arn arn:aws: iam::<111122223333>:role/
<AmazonEKSFargatePodExecutionRole> \
    --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
    --subnets subnet-<000000000000000a> subnet-<000000000000000b> subnet-
<000000000000000c>
```

2. Déclenchez le déploiement du `coredns` déploiement.

```
kubectl rollout restart -n kube-system deployment coredns
```

Étapes suivantes

- Vous pouvez commencer à migrer vos applications existantes pour les exécuter sur Fargate avec le flux suivant.
  a. the section called "Créer un profil Fargate"qui correspond à l'espace de noms Kubernetes et aux étiquettes Kubernetes de votre application.
  b. Supprimez et recréez tous les pods existants afin qu'ils soient planifiés sur Fargate. Modifiez `<namespace>` et `<deployment-type>` pour mettre à jour vos pods spécifiques.

  ```
  kubectl rollout restart -n <namespace> deployment <deployment-type>
  ```

- Déployez le the section called "Répartition de la charge des applications" pour autoriser les objets Ingress à fonctionner sur Fargate pour vos pods.
- Vous pouvez utiliser le the section called "Vertical Pod Autoscaler (VPA)" pour définir la taille initiale correcte du processeur et de la mémoire pour vos Fargate Pods, puis l'utiliser the section called "Horizontal Pod Autoscaler" pour redimensionner ces pods. Si vous souhaitez que le Vertical Pod Autoscaler redéploie automatiquement les pods vers Fargate avec des combinaisons de processeur et de mémoire plus élevées, réglez le mode du Vertical Pod Autoscaler sur ou. `Auto` `Recreate` Ceci permet de garantir une fonctionnalité correcte. Pour plus d'informations, consultez la documentation Vertical Pod Autoscaler sur GitHub.
- Vous pouvez configurer le collecteur AWS Distro pour OpenTelemetry (ADOT) pour la surveillance des applications en suivant ces instructions.

Définissez quels pods utilisent AWS Fargate lors de leur lancement

Avant de planifier des pods sur Fargate dans votre cluster, vous devez définir au moins un profil Fargate qui indique quels pods utilisent Fargate lors de leur lancement.

En tant qu'administrateur, vous pouvez utiliser un profil Fargate pour déclarer quels Pods s'exécutent sur Fargate. Vous pouvez faire cela via les sélecteurs du profil. Vous pouvez ajouter jusqu'à cinq sélecteurs à chaque profil. Chaque sélecteur doit contenir un espace de noms. Le sélecteur peut également inclure des étiquettes. Le champ de label se compose de plusieurs paires clé-valeur facultatives. Les pods qui correspondent à un sélecteur sont programmés sur Fargate. Les pods sont associés à l'aide d'un espace de noms et des étiquettes spécifiées dans le sélecteur. Si un sélecteur d'espace de noms est défini sans étiquette, Amazon EKS tente de planifier tous les pods exécutés dans cet espace de noms sur Fargate à l'aide du profil. Si un to-be-scheduled pod correspond à l'un des sélecteurs du profil Fargate, il est programmé sur Fargate.

Si un pod correspond à plusieurs profils Fargate, vous pouvez spécifier le profil utilisé par un pod en ajoutant l'étiquette Kubernetes suivante à la spécification du pod :. `eks.amazonaws.com/fargate-profile: my-fargate-profile` Le Pod doit correspondre à un sélecteur de ce profil pour être programmé sur Fargate. Les règles d'affinité/anti-affinité de Kubernetes ne s'appliquent pas et ne sont pas nécessaires avec les pods Amazon EKS Fargate.

Lorsque vous créez un profil Fargate, vous devez spécifier un rôle d'exécution du Pod. Ce rôle d'exécution concerne les composants Amazon EKS qui s'exécutent sur l'infrastructure Fargate utilisant le profil. Il est ajouté au contrôle d'accès basé sur les rôles (RBAC) Kubernetes du cluster à des fins d'autorisation. Ainsi, le `kubelet` qui est exécuté sur l'infrastructure Fargate peut s'enregistrer dans votre cluster Amazon EKS et apparaître dans votre cluster comme un nœud. Le rôle d'exécution Pod fournit également des autorisations IAM à l'infrastructure Fargate afin de permettre un accès en lecture aux référentiels d'images Amazon ECR. Pour de plus amples informations, veuillez consulter the section called “Rôle IAM d'exécution de pod”.

Les profils Fargate ne peuvent pas être modifiés. Toutefois, vous pouvez créer un nouveau profil mis à jour pour remplacer un profil existant, puis supprimer l'original.

> **Note**
>
> Tous les pods exécutés à l'aide d'un profil Fargate sont arrêtés et placés en attente lorsque le profil est supprimé.

Si tous les profils Fargate d'un cluster ont l'état `DELETING`, vous devez attendre que ce profil Fargate soit définitivement supprimé avant de pouvoir créer d'autres profils dans ce cluster.

> **Note**
>
> Fargate ne prend actuellement pas en charge Kubernetes. topologySpreadConstraints

Amazon EKS et Fargate répartissent les pods sur chacun des sous-réseaux définis dans le profil Fargate. Cependant, vous risquez de vous retrouver avec une répartition inégale. Si vous avez besoin d'une répartition uniforme, utilisez deux profils Fargate. La répartition uniforme est importante dans les scénarios où vous souhaitez déployer deux répliques sans aucune interruption de service. Nous vous recommandons de n'avoir qu'un seul sous-réseau pour chaque profil.

Composants de profil Fargate

Les composants suivants sont contenus dans un profil Fargate.

Rôle d'exécution du pod

Lorsque votre cluster crée des pods sur AWS Fargate, ceux qui s'exécutent sur `kubelet` l'infrastructure Fargate doivent effectuer des appels en votre nom. AWS APIs Par exemple, il doit effectuer des appels pour extraire des images de conteneur à partir d'Amazon ECR. Le rôle d'exécution Amazon EKS Pod fournit les autorisations IAM nécessaires à cette fin.

Lorsque vous créez un profil Fargate, vous devez spécifier un rôle d'exécution de Pod à utiliser avec vos Pods. Ce rôle est ajouté au contrôle d'accès basé sur les rôles (RBAC) Kubernetes du cluster à des fins d'autorisation. Cela permet à l'`kubelet`infrastructure Fargate de s'enregistrer auprès de votre cluster Amazon EKS et d'apparaître dans votre cluster en tant que nœud. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM d'exécution de pod".

Sous-réseaux

Les IDs sous-réseaux dans lesquels lancer des pods qui utilisent ce profil. À l'heure actuelle, aucune adresse IP publique n'est attribuée aux pods exécutés sur Fargate. Par conséquent, seuls les sous-réseaux privés sans route directe vers une passerelle Internet sont acceptés pour ce paramètre.

Sélecteurs

Les sélecteurs à sélectionner pour que les pods utilisent ce profil Fargate. Vous pouvez spécifier jusqu'à cinq sélecteurs dans un profil Fargate. Les sélecteurs intègrent les composants suivants :

- Espace de noms : vous devez spécifier un espace de noms pour un sélecteur. Le sélecteur ne correspond qu'aux pods créés dans cet espace de noms. Vous pouvez toutefois créer plusieurs sélecteurs pour cibler plusieurs espaces de noms.
- Labels : vous pouvez éventuellement spécifier des labels Kubernetes à faire correspondre pour le sélecteur. Le sélecteur ne correspond qu'aux pods qui possèdent toutes les étiquettes spécifiées dans le sélecteur.

Caractères génériques de profils Fargate

Outre les caractères autorisés par Kubernetes, vous êtes autorisé à utiliser `*` et `?` dans les critères de sélection pour les espaces de noms, les clés d'étiquette et les valeurs d'étiquette :

- `*` représente aucun, un ou plusieurs caractères. Par exemple, `prod*` peut représenter `prod` et `prod-metrics`.
- `?` représente un caractère unique (par exemple, `value?` peut représenter `valuea`). Cependant, il ne peut pas représenter `value` et `value-a`, parce que `?` ne peut représenter qu'un seul et unique caractère.

Ces caractères génériques peuvent être utilisés dans n'importe quelle position et en combinaison (par exemple, `prod*`, `*dev` et `frontend*?`). Les autres caractères génériques et formes de correspondance de modèles, tels que les expressions régulières, ne sont pas pris en charge.

S'il existe plusieurs profils correspondants pour l'espace de noms et les étiquettes dans la spécification Pod, Fargate sélectionne le profil en triant alphanumérique par nom de profil. Par exemple, si le profil A (avec le nom`beta-workload`) et le profil B (avec le nom`prod-workload`) ont des sélecteurs correspondants pour les pods à lancer, Fargate choisit le profil A `beta-workload` () pour les pods. Les pods ont des étiquettes avec le profil A sur les pods (par exemple,`eks.amazonaws.com/fargate-profile=beta-workload`).

Si vous souhaitez migrer des Fargate Pods existants vers de nouveaux profils utilisant des caractères génériques, vous pouvez procéder de deux manières :

- Créez un nouveau profil avec les sélecteurs correspondants, puis supprimez les anciens profils. Les pods étiquetés avec d'anciens profils sont reprogrammés vers de nouveaux profils correspondants.
- Si vous souhaitez migrer des charges de travail mais que vous ne savez pas quelles étiquettes Fargate figurent sur chaque Fargate Pod, vous pouvez utiliser la méthode suivante. Créez un

nouveau profil avec un nom qui trie d'abord par ordre alphanumérique parmi les profils du même cluster. Recyclez ensuite les Fargate Pods qui doivent être migrés vers de nouveaux profils.

Créer un profil Fargate

Cette section explique comment créer un profil Fargate. Vous devez également avoir créé un rôle d'exécution Pod à utiliser pour votre profil Fargate. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM d'exécution de pod". Les pods exécutés sur Fargate ne sont pris en charge que sur les sous-réseaux privés dotés d'un accès aux AWS services par passerelle NAT, mais pas de route directe vers une passerelle Internet. Cela signifie que le VPC de votre cluster doit disposer de sous-réseaux privés disponibles.

Vous pouvez créer un profil avec les éléments suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Pour créer un profil Fargate avec **`eksctl`**

Créez votre profil Fargate à l'aide de la commande `eksctl` suivante, en remplaçant chaque exemple de valeur par vos propres valeurs. Vous devez spécifier un espace de noms. Cependant, `--labels` cette option n'est pas obligatoire.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

Vous pouvez utiliser certains caractères génériques pour les étiquettes `my-kubernetes-namespace` et `key=value`. Pour de plus amples informations, veuillez consulter the section called "Caractères génériques de profils Fargate".

AWS Management Console

Pour créer un profil Fargate avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster pour lequel vous voulez créer un profil Fargate.
3. Choisissez l'onglet Calcul.
4. Sous Fargate profiles (Profils Fargate), choisissez Add Fargate profile (Ajouter un profil Fargate).
5. Sur la page Configure Fargate profile (Configurer le profil Fargate), procédez comme suit :
   a. Pour Nom, saisissez un nom unique pour votre profil Fargate, tel que `my-profile`.
   b. Pour le rôle d'exécution du Pod, choisissez le rôle d'exécution du Pod à utiliser avec votre profil Fargate. Seuls les rôles IAM avec le principal de service `eks-fargate-pods.amazonaws.com` sont affichés. Si aucun rôle n'est répertorié, vous devez en créer un. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM d'exécution de pod".
   c. Modifiez les sous-réseaux sélectionnés selon vos besoins.

      > **Note**
      > Seuls les sous-réseaux privés sont pris en charge pour les pods exécutés sur Fargate.

   d. Dans Identifications, vous pouvez éventuellement étiqueter votre profil Fargate. Ces balises ne se propagent pas aux autres ressources associées au profil, telles que les pods.
   e. Choisissez Suivant.
6. Sur la page de sélection Configurer le module, procédez comme suit :
   a. Pour Namespace, entrez un espace de noms correspondant aux pods.
      - Vous pouvez utiliser des espaces de noms spécifiques pour les faire correspondre, tels que `kube-system` ou `default`.
      - Vous pouvez utiliser certains caractères génériques (par exemple, `prod-*`) pour faire correspondre plusieurs espaces de noms (par exemple, `prod-deployment` et `prod-test`). Pour de plus amples informations, veuillez consulter the section called "Caractères génériques de profils Fargate".
   b. (Facultatif) Ajoutez des labels Kubernetes au sélecteur. Plus précisément, ajoutez-les à celui que les pods de l'espace de noms spécifié doivent correspondre.
      - Vous pouvez ajouter l'étiquette `infrastructure: fargate` au sélecteur afin que seuls les pods de l'espace de noms spécifié qui portent également l'étiquette `infrastructure: fargate` Kubernetes correspondent au sélecteur.

- Vous pouvez utiliser certains caractères génériques (par exemple, `key?: value?`) pour faire correspondre plusieurs espaces de noms (par exemple, `keya: valuea` et `keyb: valueb`). Pour de plus amples informations, veuillez consulter the section called "Caractères génériques de profils Fargate".

c. Choisissez Suivant.

7. Sur la page Vérifier et créer, vérifiez les informations de votre profil Fargate et choisissez Créer.

Supprimer un profil Fargate

Cette rubrique décrit comment supprimer un profil Fargate. Lorsque vous supprimez un profil Fargate, tous les pods planifiés sur Fargate avec le profil sont supprimés. Si ces pods correspondent à un autre profil Fargate, ils sont programmés sur Fargate avec ce profil. S'ils ne correspondent plus à aucun profil Fargate, ils ne sont pas programmés sur Fargate et peuvent rester en attente.

Un seul profil Fargate d'un cluster peut avoir le statut `DELETING` à la fois. Vous devez attendre que la suppression d'un profil Fargate soit terminée pour pouvoir supprimer un autre profil de ce cluster.

Vous pouvez supprimer un profil à l'aide de l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

`eksctl`

Supprimer un profil Fargate avec **`eksctl`**

Utilisez la commande suivante pour supprimer un profil d'un cluster. Remplacez chaque *`example value`* par vos propres valeurs.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

AWS Management Console

Supprimer un profil Fargate avec AWS Management Console

1. Ouvrez la console Amazon EKS.

2. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, sélectionnez celui dont vous souhaitez supprimer le profil Fargate.
3. Choisissez l'onglet Calcul.
4. Sélectionnez le profil Fargate à supprimer, puis l'option Delete (Supprimer).
5. Sur la page Delete Fargate profile (Supprimer le profil Fargate), saisissez le nom du profil, puis sélectionnez Delete (Supprimer).

AWS CLI

Supprimer un profil Fargate avec la CLI AWS

Utilisez la commande suivante pour supprimer un profil d'un cluster. Remplacez chaque *example value* par vos propres valeurs.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-
cluster
```

Comprendre les détails de configuration du Fargate Pod

Cette section décrit certains des détails de configuration uniques des pods pour exécuter des pods Kubernetes sur Fargate. AWS

CPU et mémoire d'un pod

Avec Kubernetes, vous pouvez définir des requêtes, une quantité minimale de vCPU et des ressources de mémoire allouées à chaque conteneur d'un Pod. Les pods sont planifiés par Kubernetes pour garantir qu'au moins les ressources demandées pour chaque pod sont disponibles sur la ressource de calcul. Pour plus d'informations, consultez Gestion des ressources de calcul des conteneurs dans la documentation Kubernetes.

> **Note**
>
> Étant donné qu'Amazon EKS Fargate n'exécute qu'un seul pod par nœud, le scénario consistant à expulser des pods en cas de diminution des ressources ne se produit pas. Tous les pods Amazon EKS Fargate fonctionnent avec une priorité garantie. Le processeur et la mémoire requis doivent donc être égaux à la limite pour tous les conteneurs. Pour

plus d'informations, consultez Configurer la qualité de service pour les pods dans la documentation Kubernetes.

Lorsque des pods sont programmés sur Fargate, les réserves de vCPU et de mémoire définies dans les spécifications du pod déterminent la quantité de processeur et de mémoire à allouer au pod.

- La demande maximale de tous les conteneurs Init est utilisée pour déterminer les besoins en vCPU et en mémoire de la demande Init.
- Les demandes de tous les conteneurs à longue durée d'exécution sont additionnées pour déterminer les besoins en vCPU et en mémoire de la demande à longue durée d'exécution.
- La plus grande des deux valeurs précédentes est choisie pour le vCPU et la demande de mémoire à utiliser pour votre Pod.
- Fargate ajoute 256 Mo à la réservation de mémoire de chaque pod pour les composants Kubernetes requis (,, et). `kubelet` `kube-proxy` `containerd`

Fargate arrondit la configuration de calcul suivante qui correspond le mieux à la somme des demandes de vCPU et de mémoire afin de garantir que les Pods disposent toujours des ressources dont ils ont besoin pour fonctionner.

Si vous ne spécifiez pas de combinaison vCPU et mémoire, la plus petite combinaison disponible est utilisée (0,25 vCPU et 0,5 Go de mémoire).

Le tableau suivant indique les combinaisons de vCPU et de mémoire disponibles pour les pods exécutés sur Fargate.

Valeur vCPU	Valeur de mémoire
0,25 vCPU	0,5 Go, 1 Go, 2 Go
0,5 vCPU	1 Go, 2 Go, 3 Go, 4 Go
1 vCPU	2 Go, 3 Go, 4 Go, 5 Go, 6 Go, 7 Go, 8 Go
2 vCPU	Entre 4 Go et 16 Go par incrément de 1 Go
4 vCPU	Entre 8 Go et 30 Go par incrément de 1 Go

Valeur vCPU	Valeur de mémoire
8 vCPU	Entre 16 Go et 60 Go par incréments de 4 Go
16 vCPU	Entre 32 Go et 120 Go par incréments de 8 Go

La mémoire supplémentaire réservée aux composants Kubernetes peut entraîner le provisionnement d'une tâche Fargate avec plus de v que ce qui est demandé. CPUs Par exemple, une demande pour 1 vCPU et 8 Go de mémoire sera ajoutée à sa demande de mémoire, et fournira à une tâche Fargate 2 V et 9 Go de mémoire, car aucune tâche avec 1 vCPU CPUs et 9 Go de mémoire n'est disponible.

Il n'y a aucune corrélation entre la taille du Pod exécuté sur Fargate et la taille du nœud signalée par Kubernetes avec. `kubectl get nodes` La taille du nœud signalée est souvent supérieure à la capacité du Pod. Vous pouvez vérifier la capacité du Pod à l'aide de la commande suivante. *`default`*Remplacez-le par l'espace de noms de votre Pod et *`pod-name`* par le nom de votre Pod.

```
kubectl describe pod --namespace default pod-name
```

L'exemple qui suit illustre un résultat.

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

L'`CapacityProvisioned`annotation représente la capacité imposée du pod et détermine le coût de fonctionnement de votre pod sur Fargate. Pour plus d'informations sur la tarification de ces configurations de calcul, reportez-vous à Tarification AWS.

Stockage Fargate

Un Pod exécuté sur Fargate monte automatiquement un système de fichiers Amazon EFS, sans nécessiter d'étapes d'installation manuelle du pilote. Vous ne pouvez pas utiliser le provisionnement dynamique de volumes persistants avec les nœuds Fargate, mais vous pouvez utiliser le provisionnement statique. Pour plus d'informations, consultez Amazon EFS CSI Driver activé GitHub.

Lors du provisionnement, chaque Pod exécuté sur Fargate reçoit 20 GiB de stockage éphémère par défaut. Ce type de stockage est supprimé après l'arrêt d'un pod. Le chiffrement du volume de stockage éphémère est activé par défaut pour les nouveaux pods lancés sur Fargate. Le stockage

éphémère du Pod est crypté à l'aide d'un algorithme de cryptage AES-256 utilisant des clés gérées par AWS Fargate.

> **Note**
>
> Le stockage utilisable par défaut pour les pods Amazon EKS exécutés sur Fargate est inférieur à 20 GiB. Cela est dû au fait qu'une partie de l'espace est utilisée par les modules Kubernetes `kubelet` et les autres modules qui sont chargés dans le Pod.

Vous pouvez augmenter la capacité totale de stockage éphémère jusqu'à un maximum de 175 GiB. Pour configurer la taille avec Kubernetes, spécifiez les demandes de `ephemeral-storage` ressources pour chaque conteneur d'un Pod. Lorsque Kubernetes planifie des pods, il garantit que la somme des demandes de ressources pour chaque pod est inférieure à la capacité de la tâche Fargate. Pour plus d'informations, consultez la section Gestion des ressources pour les pods et les conteneurs dans la documentation de Kubernetes.

Amazon EKS Fargate alloue une capacité de stockage éphémère supérieure à celle demandée aux fins de l'utilisation du système. Par exemple, une demande de 100 GiB allouera à une tâche Fargate une capacité de stockage éphémère de 115 GiB.

Définir des actions pour les événements de mise à AWS jour du système d'exploitation Fargate

Amazon EKS applique régulièrement des correctifs au système d'exploitation des nœuds AWS Fargate afin de garantir leur sécurité. Dans le cadre du processus d'application des correctifs, nous recyclons les nœuds pour installer les correctifs du système d'exploitation. Les tentatives de mises à jour sont réalisées de manière à avoir des répercussions minimales sur vos services. Cependant, si les pods ne sont pas expulsés avec succès, ils doivent parfois être supprimés. Voici les actions que vous pouvez effectuer pour minimiser les perturbations potentielles :

- Définissez les budgets d'interruption des pods appropriés (PDBs) pour contrôler le nombre de pods qui sont en panne simultanément.
- Créez des EventBridge règles Amazon pour gérer les expulsions ratées avant que les Pods ne soient supprimés.
- Redémarrez manuellement vos pods concernés avant la date d'expulsion indiquée dans la notification que vous recevez.
- Créez une configuration de notification dans Notifications AWS utilisateur.

Amazon EKS travaille en étroite collaboration avec la communauté Kubernetes pour mettre à disposition les correctifs de bogues et les correctifs de sécurité le plus rapidement possible. Tous les Fargate Pods démarrent avec la dernière version du correctif Kubernetes, disponible sur Amazon EKS pour la version Kubernetes de votre cluster. Si vous possédez un Pod doté d'une ancienne version de correctif, Amazon EKS peut le recycler pour le mettre à jour vers la dernière version. Cela garantit que vos Pods sont équipés des dernières mises à jour de sécurité. Ainsi, en cas de problème critique lié aux vulnérabilités et aux expositions communes (CVE), vous êtes tenu au courant afin de réduire les risques de sécurité.

Lorsque le système AWS d'exploitation Fargate est mis à jour, Amazon EKS vous envoie une notification indiquant les ressources concernées et la date des prochaines expulsions des pods. Si la date d'expulsion indiquée ne vous convient pas, vous avez la possibilité de redémarrer manuellement les pods concernés avant la date d'expulsion indiquée dans la notification. Tous les pods créés avant le moment où vous recevez la notification sont susceptibles d'être expulsés. Reportez-vous à la documentation Kubernetes pour obtenir des instructions supplémentaires sur le redémarrage manuel de vos pods.

Pour limiter le nombre de pods qui sont hors service en même temps lorsque les pods sont recyclés, vous pouvez définir des budgets d'interruption des pods (PDBs). Vous pouvez l'utiliser PDBs pour définir une disponibilité minimale en fonction des exigences de chacune de vos applications tout en autorisant les mises à jour. La disponibilité minimale de votre PDB doit être inférieure à 100 %. Pour plus d'informations, consultez la section Spécification d'un budget d'interruption pour votre application dans la documentation de Kubernetes.

Amazon EKS utilise l'API Eviction pour vider le Pod en toute sécurité tout en respectant les paramètres PDBs que vous avez définis pour l'application. Les pods sont expulsés par zone de disponibilité pour minimiser les répercussions. Si l'expulsion aboutit, le nouveau Pod reçoit le dernier correctif et aucune autre action n'est requise.

Lorsque l'expulsion d'un pod échoue, Amazon EKS envoie un événement à votre compte avec des informations sur les pods dont l'expulsion a échoué. Vous pouvez agir en fonction du message avant l'heure de fin prévue. Le temps spécifique varie en fonction de l'urgence du correctif. Le moment venu, Amazon EKS tente à nouveau d'expulser les Pods. Toutefois, si l'expulsion échoue lors de cette tentative, un nouvel événement n'est pas envoyé. Si l'expulsion échoue à nouveau, vos pods existants sont régulièrement supprimés afin que les nouveaux pods puissent bénéficier du dernier correctif.

Voici un exemple d'événement reçu lorsque l'expulsion du Pod échoue. Il contient des détails sur le cluster, le nom du pod, l'espace de noms du pod, le profil Fargate et l'heure de fin prévue.

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

De plus, le fait d'en avoir plusieurs PDBs associés à un pod peut provoquer un échec d'expulsion. Cet événement renvoie le message d'erreur suivant.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

Vous pouvez créer une action souhaitée en fonction de cet événement. Par exemple, vous pouvez ajuster le budget d'interruption de votre pod (PDB) pour contrôler la manière dont les pods sont expulsés. Plus précisément, supposons que vous commenciez avec un PDB qui spécifie le pourcentage cible de pods disponibles. Avant que vos pods ne soient résiliés de force lors d'une mise à niveau, vous pouvez ajuster le PDB à un pourcentage différent de pods. Pour recevoir cet événement, vous devez créer une EventBridge règle Amazon dans le AWS compte et AWS la région auxquels appartient le cluster. La règle doit utiliser le modèle personnalisé suivant. Pour plus d'informations, consultez la section Création de EventBridge règles Amazon qui réagissent aux événements dans le guide de EventBridge l'utilisateur Amazon.

```
{
  "source": ["aws.eks"],
```

```
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

Une cible appropriée peut être définie pour que l'événement le capture. Pour obtenir la liste complète des cibles disponibles, consultez les EventBridge cibles Amazon dans le guide de EventBridge l'utilisateur Amazon. Vous pouvez également créer une configuration de notification dans Notifications AWS utilisateur. Lorsque vous utilisez le AWS Management Console pour créer la notification, sous Règles relatives aux événements, choisissez Elastic Kubernetes Service (EKS) pour le nom du service et EKS Fargate AWS Pod Scheduled Termination pour le type d'événement. Pour plus d'informations, voir Commencer à utiliser les notifications AWS utilisateur dans le Guide de AWS l'utilisateur des notifications utilisateur.

Voir FAQs: Avis d'expulsion de Fargate Pod AWS dans Re:post pour les questions fréquemment posées concernant les expulsions d'EKS Pod.

Collectez l' AWS application Fargate et les statistiques d'utilisation

Vous pouvez collecter des métriques du système et des métriques CloudWatch d'utilisation pour AWS Fargate.

Métriques d'application

Pour les applications exécutées sur Amazon EKS et AWS Fargate, vous pouvez utiliser Distro OpenTelemetry for (AWS ADOT). ADOT vous permet de collecter les métriques du système et de les envoyer aux tableaux de bord de CloudWatch Container Insights. Pour démarrer avec ADOT pour les applications exécutées sur Fargate, consultez la section CloudWatch Utilisation de Container Insights AWS with Distro OpenTelemetry dans la documentation ADOT.

Métriques d'utilisation

Vous pouvez utiliser les statistiques CloudWatch d'utilisation pour obtenir une visibilité sur l'utilisation des ressources par votre compte. Utilisez ces indicateurs pour visualiser l'utilisation actuelle de vos services sur CloudWatch des graphiques et des tableaux de bord.

AWS Les métriques d'utilisation de Fargate correspondent aux AWS quotas de service. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche un quota de service. Pour de plus amples informations sur les Service Quotas pour Fargate, consultez the section called "Quotas de service".

AWS Fargate publie les métriques suivantes dans l'espace de noms. `AWS/Usage`

Métrique	Description
`ResourceCount`	Nombre total des ressources spécifiées exécutées sur votre compte. La ressource est définie par les dimensions associées à la métrique.

Les dimensions suivantes sont utilisées pour affiner les métriques d'utilisation publiées par AWS Fargate.

Dimension	Description
`Service`	Nom du AWS service contenant la ressource . Pour les métriques AWS d'utilisation de Fargate, la valeur de cette dimension est. `Fargate`
`Type`	Type d'entité signalée. Actuellement, la seule valeur valide pour les métriques d' AWS utilisati on de Fargate est. `Resource`
`Resource`	Type de ressource en cours d'exécution. Actuellement, AWS Fargate renvoie des informations sur votre utilisation de Fargate On-Demand. La valeur de ressource pour l'utilisa tion de Fargate On-Demand est `OnDemand`. [REMARQUE] ==== L'utilisation de Fargate On-Demand combine les pods Amazon EKS utilisant Fargate, les tâches Amazon ECS utilisant le type de lancement Fargate et les tâches Amazon ECS utilisant le fournisseur de capacité. `FARGATE` ====

Dimension	Description
`Class`	Classe de ressource suivie. Actuellement, AWS Fargate n'utilise pas la dimension de classe.

Création d'une CloudWatch alarme pour surveiller les métriques d'utilisation des ressources de Fargate

AWS Fargate CloudWatch fournit des métriques d'utilisation qui correspondent aux quotas de service pour AWS l'utilisation des ressources Fargate On-Demand. Dans la console Service Quotas, vous pouvez visualiser votre utilisation sur un graphique. Vous pouvez également configurer des alarmes qui vous alertent lorsque votre utilisation approche d'un quota de service. Pour de plus amples informations, veuillez consulter the section called "Collectez des métriques".

Suivez les étapes ci-dessous pour créer une CloudWatch alarme basée sur les métriques d'utilisation des ressources de Fargate.

1. Ouvrez la console Service Quotas sur https://console.aws.amazon.com/servicequotas/.
2. Dans le volet de navigation de gauche, sélectionnez AWS services.
3. Dans la liste des AWS services, recherchez et sélectionnez AWS Fargate.
4. Dans la liste Quotas de service, sélectionnez le quota d'utilisation Fargate pour lequel vous souhaitez créer une alarme.
5. Dans la section CloudWatch Alarmes Amazon, choisissez Create.
6. Pour Alarm threshold (Seuil d'alarme), choisissez le pourcentage de la valeur de quota appliquée que vous souhaitez définir comme valeur d'alarme.
7. Pour Nom de l'alarme, saisissez un nom pour l'alarme, puis choisissez Créer.

Démarrez AWS la journalisation Fargate pour votre cluster

Amazon EKS on Fargate propose un routeur de journal intégré basé sur Fluent Bit. Cela signifie que vous n'exécutez pas explicitement un conteneur Fluent Bit en tant que sidecar, mais qu'Amazon le gère pour vous. Tout ce que vous avez à faire est de configurer le routeur de journaux. La configuration se fait par le biais d'un `ConfigMap` qui doit répondre aux critères suivants :

- Nommé `aws-logging`
- Créé dans un espace de noms dédié appelé `aws-observability`

- Ne peut pas dépasser 5 300 caractères.

Une fois que vous l'avez créé`ConfigMap`, Amazon EKS on Fargate le détecte automatiquement et configure le routeur de journaux avec celui-ci. Fargate utilise une version AWS de for Fluent Bit, une distribution conforme en amont de Fluent Bit gérée par. AWS Pour plus d'informations, voir AWS Fluent Bit activé GitHub.

Le routeur de journaux vous permet d'utiliser l'ensemble des services proposés AWS pour l'analyse et le stockage des journaux. Vous pouvez diffuser les journaux de Fargate directement vers Amazon CloudWatch, Amazon Service. OpenSearch Vous pouvez également diffuser des journaux vers des destinations telles qu'Amazon S3, Amazon Kinesis Data Streams et des outils partenaires via Amazon Data Firehose.

- Un profil Fargate existant qui spécifie un espace de noms Kubernetes existant dans lequel vous déployez des Fargate Pods. Pour de plus amples informations, veuillez consulter the section called "Étape 3 : Créez un profil Fargate pour votre cluster".
- Un rôle d'exécution de Fargate Pod existant. Pour de plus amples informations, veuillez consulter the section called "Étape 2 : Création d'un rôle d'exécution Fargate Pod".

Configuration du routeur de journaux

> **Important**
>
> Pour que les journaux soient publiés correctement, il doit y avoir un accès réseau entre le VPC dans lequel se trouve votre cluster et la destination du journal. Cela concerne principalement les utilisateurs qui personnalisent les règles de sortie pour leur VPC. Pour un exemple d'utilisation CloudWatch, consultez la section Utilisation CloudWatch des journaux avec des points de terminaison VPC d'interface dans le guide de l'utilisateur Amazon CloudWatch Logs.

Dans les étapes suivantes, remplacez chaque *`example value`* par vos propres valeurs.

1. Créez un espace de noms Kubernetes dédié nommé `aws-observability`.
   a. Enregistrez le contenu suivant dans un fichier nommé `aws-observability-namespace.yaml` sur votre ordinateur. La valeur pour `name` doit être `aws-observability` et le label `aws-observability: enabled` est obligatoire.

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

b. Créez l'espace de noms.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Créez un `ConfigMap` avec une valeur de données `Fluent Conf` pour envoyer les journaux des conteneurs vers une destination. Fluent Conf est Fluent Bit, un langage de configuration de processeur de journaux rapide et léger utilisé pour acheminer les journaux des conteneurs vers la destination de journal de votre choix. Pour plus d'informations, consultez Fichier de configuration dans la documentation Fluent Bit.

> **Important**
>
> Dans une `Fluent Conf` type, les principales sections incluses sont `Service`, `Input`, `Filter` et `Output`. Le routeur de journaux Fargate n'accepte cependant que :
>
> - Les sections `Filter` et `Output`.
> - Une section `Parser`.
>
> Si vous fournissez d'autres sections, elles seront rejetées.

Le routeur de journal Fargate gère les sections `Service` et `Input`. Il contient la `Input` section suivante, qui ne peut pas être modifiée et n'est pas nécessaire dans votre`ConfigMap`. Cependant, vous pouvez en tirer des informations, telles que la limite de la mémoire tampon et la balise appliquée pour les journaux.

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
```

```
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

Lors de la création du `ConfigMap`, prenez en compte les règles suivantes que Fargate utilise pour valider les champs :

- `[FILTER]`, `[OUTPUT]` et `[PARSER]` sont censés être spécifiés sous chaque clé correspondante. Par exemple, `[FILTER]` doit être inférieur à `filters.conf`. Vous pouvez avoir un ou plusieurs `[FILTER]` sous `filters.conf`. `[OUTPUT]` et `[PARSER]` doivent également être sous leurs clés correspondantes. En spécifiant plusieurs sections `[OUTPUT]`, vous pouvez acheminer vos journaux vers différentes destinations en même temps.
- Fargate valide les clés requises de chaque section. `Name` et `match` sont nécessaires pour chaque `[FILTER]` et `[OUTPUT]`. `Name` et `format` sont nécessaires pour chaque `[PARSER]`. Ces noms sont sensibles à la casse.
- Les variables d'environnement telles que `${ENV_VAR}` celles qui ne sont pas autorisées dans le`ConfigMap`.
- L'indentation doit être la même pour la directive ou la paire clé-valeur dans chaque `filters.conf`, `output.conf` et `parsers.conf`. Les paires clé-valeur doivent être indentées plus que les directives.
- Fargate valide par rapport aux filtres pris en charge suivants : `grep`, `parser`, `record_modifier`, `rewrite_tag`, `throttle`, `nest`, `modify` et `kubernetes`.
- Fargate valide par rapport à la sortie prise en charge suivante : `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs` et `kinesis`.
- Au moins un `Output` plugin compatible doit être fourni dans le `ConfigMap` pour activer la journalisation. `Filter`et `Parser` ne sont pas tenus d'activer la journalisation.

  Vous pouvez également exécuter Fluent Bit sur Amazon EC2 en utilisant la configuration souhaitée pour résoudre les problèmes liés à la validation. Créez votre `ConfigMap` en utilisant l'un des exemples suivants.

  > **Important**
  >
  > La journalisation Amazon EKS Fargate ne prend pas en charge la configuration dynamique d'un. `ConfigMap` Toute modification apportée à a `ConfigMap` s'applique

uniquement aux nouveaux pods. Les modifications ne sont pas appliquées aux pods existants.

Créez un `ConfigMap` en utilisant l'exemple pour votre destination de journal désirée.

> **Note**
>
> Vous pouvez également utiliser Amazon Kinesis Data Streams comme destination du journal. Si vous utilisez Kinesis Data Streams, assurez-vous que l'autorisation `kinesis:PutRecords` a été accordée au rôle d'exécution du pod. Pour plus d'informations, consultez les autorisations Amazon Kinesis Data Streams dans le Fluent Bit : Official Manual.

Example

CloudWatch

Deux options de sortie s'offrent à vous lorsque vous utilisez CloudWatch :

- Un plugin de sortie écrit en C
- Un plugin de sortie écrit en Golang

L'exemple suivant montre comment utiliser le `cloudwatch_logs` plugin pour envoyer des journaux à CloudWatch.

a. Enregistrez le contenu suivant dans un fichier nommé `aws-logging-cloudwatch-configmap.yaml`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Les paramètres sous `[OUTPUT]` sont requises.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
```

```
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

Si vous souhaitez envoyer des journaux à Amazon OpenSearch Service, vous pouvez utiliser es output, qui est un plugin écrit en C. L'exemple suivant montre comment utiliser le plug-in pour envoyer des journaux à OpenSearch.

a. Enregistrez le contenu suivant dans un fichier nommé `aws-logging-opensearch-configmap.yaml`. Remplacez chaque *`example value`* par vos propres valeurs.

```
kind: ConfigMap
apiVersion: v1
```

```
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

Deux options de sortie s'offrent à vous lorsque vous envoyez des logs à Firehose :

- kinesis_firehose — Un plugin de sortie écrit en C.
- firehose — Un plugin de sortie écrit en Golang.

L'exemple suivant vous montre comment utiliser le `kinesis_firehose` plugin pour envoyer des logs à Firehose.

a. Enregistrez le contenu suivant dans un fichier nommé `aws-logging-firehose-configmap.yaml`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
```

```
    Name  kinesis_firehose
    Match *
    region region-code
    delivery_stream my-stream-firehose
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Configurez des autorisations pour le rôle d'exécution Fargate Pod afin d'envoyer les journaux vers votre destination.

a. Téléchargez la politique IAM de votre destination sur votre ordinateur.

Example

CloudWatch

Téléchargez la politique CloudWatch IAM sur votre ordinateur. Vous pouvez également consulter la politique sur GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Téléchargez la politique OpenSearch IAM sur votre ordinateur. Vous pouvez également consulter la politique sur GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

Assurez-vous que le contrôle d'accès OpenSearch des tableaux de bord est correctement configuré. `all_access role`Dans les OpenSearch tableaux de bord, le rôle d'exécution Fargate Pod et le rôle IAM doivent être mappés. Le même mappage doit être fait pour le rôle `security_manager`. Vous pouvez ajouter les mappages précédents en sélectionnant `Menu`, `Security` et `Roles`, puis sélectionner les rôles correspondants. Pour plus d'informations, consultez Comment résoudre les problèmes liés aux CloudWatch journaux afin qu'ils soient diffusés sur mon domaine Amazon ES ? .

Firehose

Téléchargez la politique IAM de Firehose sur votre ordinateur. Vous pouvez également consulter la politique sur GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

b. Créez une stratégie IAM à partir du fichier de stratégie que vous avez téléchargé.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

c. Attachez la politique IAM au rôle d'exécution de pod spécifié pour votre profil Fargate avec la commande suivante. Remplacez *111122223333* par votre ID de compte. *AmazonEKSFargatePodExecutionRole*Remplacez-le par le rôle d'exécution de votre Pod (pour plus d'informations, voirthe section called "Étape 2 : Création d'un rôle d'exécution Fargate Pod").

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Prise en charge des filtres Kubernetes

Le filtre Kubernetes de Fluent Bit vous permet d'ajouter des métadonnées Kubernetes à vos fichiers journaux. Pour plus d'informations sur le filtre, consultez Kubernetes dans la documentation Fluent Bit. Vous pouvez appliquer un filtre en utilisant le point de terminaison du serveur d'API.

```
filters.conf: |
    [FILTER]
        Name             kubernetes
        Match            kube.*
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

> **Important**
>
> - `Kube_URL`, `Kube_CA_File`, `Kube_Token_Command` et `Kube_Token_File` sont des paramètres de configuration appartenant au service et ne doivent pas être spécifiés. Amazon EKS Fargate remplit ces valeurs.
> - `Kube_Meta_Cache_TTL` est le moment où Fluent Bit attend qu'il communique avec le serveur d'API pour obtenir les dernières métadonnées. Si `Kube_Meta_Cache_TTL` ce n'est pas spécifié, Amazon EKS Fargate ajoute une valeur par défaut de 30 minutes afin de réduire la charge sur le serveur d'API.

Pour envoyer les journaux de processus Fluent Bit à votre compte

Vous pouvez éventuellement envoyer les journaux de traitement Fluent Bit à Amazon CloudWatch en utilisant les méthodes suivantes`ConfigMap`. L'expédition des journaux de traitement Fluent Bit CloudWatch nécessite des frais d'ingestion et de stockage supplémentaires. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

Les journaux se trouvent CloudWatch dans la même AWS région que le cluster. Le nom du groupe de journaux est `my-cluster-fluent-bit-logs` et le nom du flux de journaux Fluent Bit est `fluent-bit-podname-pod-namespace` .

> **Note**
>
> - Les journaux de processus sont envoyés seulement lorsque le processus Fluent Bit démarre avec succès. S'il y a un échec lors du démarrage de Fluent Bit, les journaux de processus sont manqués. Vous ne pouvez expédier les journaux de processus qu'à CloudWatch.
> - Pour déboguer l'envoi des journaux de processus à votre compte, vous pouvez appliquer le `ConfigMap` précédent pour obtenir les journaux de processus. L'échec du démarrage de Fluent Bit est généralement dû au fait que votre `ConfigMap` n'est pas analysé ou accepté par Fluent Bit lors du démarrage.

Pour arrêter d'expédier les journaux de processus Fluent Bit

L'expédition des journaux de traitement Fluent Bit CloudWatch nécessite des frais d'ingestion et de stockage supplémentaires. Pour exclure les journaux de processus d'une configuration de `ConfigMap` existante, procédez comme suit.

1. Localisez le groupe de CloudWatch journaux créé automatiquement pour les journaux de processus Fluent Bit de votre cluster Amazon EKS après avoir activé la journalisation Fargate. Il utilise le format `my-cluster-fluent-bit-logs`.
2. Supprimez les flux de CloudWatch journaux existants créés pour les journaux de processus de chaque pod dans le groupe de CloudWatch journaux.
3. Modifiez la `ConfigMap` et définissez `flb_log_cw: "false"`.
4. Redémarrez tous les pods existants dans le cluster.

Tester l'application

1. Déployez un exemple de pod.
   a. Enregistrez le contenu suivant dans un fichier nommé `sample-app.yaml` sur votre ordinateur.

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

b. Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f sample-app.yaml
```

2. Affichez les journaux NGINX en utilisant les destinations que vous avez configurées dans le fichier `ConfigMap`.

Considérations sur les tailles

Nous vous suggérons de prévoir jusqu'à 50 Mo de mémoire pour le routeur de journaux. Si votre application doit générer des journaux à un débit très élevé, vous devez prévoir jusqu'à 100 Mo.

Résolution des problèmes

Pour vérifier si la fonctionnalité de journalisation est activée ou désactivée pour une raison quelconque, par exemple si elle n'est pas valide`ConfigMap`, et pourquoi elle n'est pas valide, vérifiez les événements de votre Pod auprès de`kubectl describe pod` *`pod-name`* . La sortie peut inclure des événements Pod qui précisent si la journalisation est activée ou non, comme l'exemple de sortie suivant.

```
[...]
```

```
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
[...]
Events:
  Type     Reason           Age        From
               Message
  ----     ------           ----       ----
               -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
                Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Les événements du Pod sont éphémères et leur durée dépend des paramètres. Vous pouvez également afficher les annotations d'un pod à l'aide `kubectl describe pod` *`pod-name`* de. Dans l'annotation Pod, vous trouverez des informations indiquant si la fonction de journalisation est activée ou désactivée et pour quelle raison.

Choisissez un type d'instance de EC2 nœud Amazon optimal

Amazon EC2 propose une large sélection de types d'instances pour les nœuds de travail. Chaque type d'instance offre des capacités de calcul, de mémoire, de stockage et de réseau différentes. Chaque instance est également regroupée dans une famille d'instances en fonction de ces capacités. Pour obtenir une liste, consultez la section Types d'instances disponibles dans le guide de EC2 l'utilisateur Amazon. Amazon EKS publie plusieurs variantes d'Amazon EC2 AMIs pour permettre le support. Pour vous assurer que le type d'instance que vous sélectionnez est compatible avec Amazon EKS, tenez compte des critères suivants.

- Tous les Amazon EKS AMIs ne prennent actuellement pas en charge `mac` cette famille.
- Arm et Amazon EKS non accéléré AMIs ne prennent pas en charge les`g3`, `g4inf`, et `p` les familles.
- Amazon EKS accéléré AMIs ne prend pas en charge les `ac`,`hpc`,`m`, et `t` les familles.
- Pour les instances basées sur ARM, Amazon Linux 2023 (AL2023) ne prend en charge que les types d'instances qui utilisent des processeurs Graviton2 ou version ultérieure. AL2023 ne prend pas en charge les `A1` instances.

Lorsque vous choisissez entre les types d'instances pris en charge par Amazon EKS, tenez compte des fonctionnalités suivantes de chaque type.

Nombre d'instances dans un groupe de nœuds

En général, il est préférable d'utiliser des instances moins nombreuses et plus volumineuses, en particulier si vous avez un grand nombre de daemonsets. Chaque instance nécessitant des appels API vers le serveur d'API, plus le nombre d'instances est élevé, plus la charge sur le serveur d'API est importante.

Système d'exploitation

Examinez les types d'instances pris en charge pour Linux, Windows et Bottlerocket. Avant de créer des instances Windows, consultez Déployer des nœuds Windows sur des clusters EKS.

Architecture matérielle

Avez-vous besoin de x86 ou d'Arm ? Avant de déployer des instances Arm, consultez Arm Amazon Linux optimisé pour Amazon EKS AMIs. Avez-vous besoin d'instances basées sur le système Nitro (Linux ou Windows) ou dotées de fonctionnalités accélérées ? Si vous avez besoin de fonctionnalités accélérées, vous ne pouvez utiliser Linux qu'avec Amazon EKS.

Nombre maximum de pods

Étant donné que chaque pod se voit attribuer sa propre adresse IP, le nombre d'adresses IP prises en charge par un type d'instance est un facteur déterminant pour déterminer le nombre de pods qui peuvent s'exécuter sur l'instance. Pour déterminer manuellement le nombre de pods pris en charge par un type d'instance, consultezthe section called "Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon".

> **Note**
>
> Si vous utilisez une AMI Amazon Linux 2 optimisée pour Amazon EKS, `v20220406` ou une version plus récente, vous pouvez utiliser un nouveau type d'instance sans passer à la dernière AMI. Pour ceux-ci AMIs, l'AMI calcule automatiquement la `max-pods` valeur nécessaire si elle n'est pas répertoriée dans le eni-max-podsfichier .txt. Les types d'instance actuellement en prévisualisation peuvent ne pas être pris en charge par Amazon EKS par défaut. Les valeurs pour `max-pods` pour de tels types doivent encore être ajoutées à `eni-max-pods.txt` dans notre AMI.

AWS Les types d'instances Nitro System prennent éventuellement en charge un plus grand nombre d'adresses IP que les types d'instances autres que Nitro System. Cependant, les

adresses IP attribuées à une instance ne sont pas toutes accessibles aux Pods. Pour attribuer un nombre significativement plus élevé d'adresses IP à vos instances, la version `1.9.0` ou ultérieure du module complémentaire Amazon VPC CNI doit être installée dans votre cluster et configurée de manière appropriée. Pour de plus amples informations, veuillez consulter the section called "Augmenter le nombre d'adresses IP". Pour attribuer le plus grand nombre d'adresses IP à vos instances, vous devez avoir installé la version `1.10.1` ou une version ultérieure du module complémentaire Amazon VPC CNI dans votre cluster et déployer le cluster avec la famille `IPv6`.

Famille IP

Vous pouvez utiliser n'importe quel type d'instance pris en charge lorsque vous utilisez la `IPv4` famille d'un cluster, ce qui permet à votre cluster d'attribuer des `IPv4` adresses privées à vos pods et services. Mais si vous souhaitez utiliser la famille `IPv6` pour votre cluster, alors vous devez utiliser les types d'instance AWS Nitro System ou les types d'instance matériel nu. Seul `IPv4` est pris en charge pour les instances Windows. Votre cluster doit utiliser la version `1.10.1` ou une version ultérieure du module complémentaire Amazon VPC CNI. Pour plus d'informations sur l'utilisation de `IPv6`, consultez the section called "IPv6".

Version du module complémentaire Amazon VPC CNI que vous utilisez

La dernière version du plug-in CNI Amazon VPC pour Kubernetes prend en charge ces types d'instance. Il peut être nécessaire de mettre à jour la version de votre module complémentaire CNI Amazon VPC pour bénéficier des derniers types d'instance pris en charge. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC". La dernière version prend en charge les dernières fonctions pour une utilisation avec Amazon EKS. Les versions antérieures ne prennent pas en charge toutes les fonctionnalités. Vous pouvez afficher les fonctions prises en charge par les différentes versions dans Changelog sur GitHub.

AWS Région dans laquelle vous créez vos nœuds

Tous les types d'instances ne sont pas disponibles dans toutes les AWS régions.

Si vous utilisez des groupes de sécurité pour les pods

Si vous utilisez des groupes de sécurité pour Pods, seuls des types d'instances spécifiques sont pris en charge. Pour de plus amples informations, veuillez consulter the section called "Groupes de sécurité pour Pods".

Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon

Étant donné que chaque pod se voit attribuer sa propre adresse IP, le nombre d'adresses IP prises en charge par un type d'instance est un facteur déterminant pour déterminer le nombre de pods qui peuvent s'exécuter sur l'instance. Amazon EKS fournit un script que vous pouvez télécharger et exécuter pour déterminer le nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance. Le script utilise les attributs matériels de chaque instance et les options de configuration pour déterminer le nombre maximum de pods. Vous pouvez utiliser le numéro renvoyé lors de ces étapes pour activer des fonctionnalités telles que l'attribution d'adresses IP à des pods à partir d'un sous-réseau différent de celui de l'instance et l'augmentation significative du nombre d'adresses IP pour votre instance. Si vous utilisez un groupe de nœuds géré avec plusieurs types d'instances, utilisez une valeur qui fonctionne pour tous les types d'instances.

1. Téléchargez un script que vous pouvez utiliser pour calculer le nombre maximum de pods pour chaque type d'instance.

   ```
   curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/
   al2/runtime/max-pods-calculator.sh
   ```

2. Marquez le script comme exécutable sur votre ordinateur.

   ```
   chmod +x max-pods-calculator.sh
   ```

3. Exécutez le script en remplaçant *`m5.large`* par le type d'instance que vous prévoyez de déployer et *`1.9.0-eksbuild.1`* par votre version du module complémentaire CNI Amazon VPC. Pour déterminer la version de votre module complémentaire, consultez les procédures de mise à jour dans Assign IPs to Pods with the Amazon VPC CNI.

   ```
   ./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
   ```

   L'exemple qui suit illustre un résultat.

   ```
   29
   ```

   Vous pouvez ajouter les options suivantes au script pour connaître le nombre maximum de pods pris en charge lors de l'utilisation des fonctionnalités facultatives.

- `--cni-custom-networking-enabled`— Utilisez cette option lorsque vous souhaitez attribuer des adresses IP provenant d'un sous-réseau différent de celui de votre instance. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée". L'ajout de cette option au script précédent avec les mêmes valeurs d'exemple donne `20`.
- `--cni-prefix-delegation-enabled` : utilisez cette option lorsque vous souhaitez attribuer beaucoup plus d'adresses IP à chaque interface réseau Elastic. Cette fonctionnalité nécessite une instance Amazon Linux qui fonctionne sur le système Nitro et la version `1.9.0` ou ultérieure du module complémentaire CNI Amazon VPC. Pour de plus amples informations, veuillez consulter the section called "Augmenter le nombre d'adresses IP". L'ajout de cette option au script précédent avec les mêmes valeurs d'exemple donne `110`.

Vous pouvez également exécuter le script avec l'option `--help` permettant de voir toutes les options disponibles.

> **Note**
>
> Le script de calcul Max Pods limite la valeur de retour en `110` fonction des seuils d'évolutivité de Kubernetes et des paramètres recommandés. Si votre type d'instance est supérieur à 30 VCPUs, cette limite passe à`250`, un chiffre basé sur les tests internes de l'équipe d'évolutivité Amazon EKS. Pour plus d'informations, consultez l'article de blog Plugin CNI Amazon VPC augmente les limites du nombre de pods par nœud (français non garanti).

Considérations relatives au mode automatique EKS

Le mode automatique EKS limite le nombre de pods sur les nœuds à la valeur la plus faible des valeurs suivantes :

- Capuchon rigide 110 dosettes
- Le résultat du calcul du nombre maximum de capsules décrit ci-dessus.

Créez des nœuds avec des images optimisées prédéfinies

Vous pouvez déployer des nœuds avec des Amazon Machine Images (AMIs) prédéfinies optimisées pour Amazon EKS ou des images personnalisées AMIs lorsque vous utilisez des groupes de nœuds

gérés ou des nœuds autogérés. Si vous utilisez des nœuds hybrides, consultezthe section called "Préparer le système d'exploitation". Pour plus d'informations sur chaque type d'AMI optimisées pour Amazon EKS, veuillez consulter l'une des rubriques suivantes. Pour obtenir des instructions sur la façon de créer votre propre AMI personnalisée, veuillez consulter the section called "Constructions personnalisées".

Avec le mode automatique d'Amazon EKS, EKS gère l' EC2 instance, notamment en sélectionnant et en mettant à jour l'AMI.

Rubriques

- Guide des fonctionnalités d' AL2 EKS et de AMIs transition AL2 accélérée
- Créez des nœuds avec Amazon Linux optimisé AMIs
- Créez des nœuds avec Bottlerocket optimisé AMIs
- Créez des nœuds avec Ubuntu Linux optimisé AMIs
- Créez des nœuds avec Windows optimisé AMIs

Guide des fonctionnalités d' AL2 EKS et de AMIs transition AL2 accélérée

AWS mettra fin à la prise en charge d'EKS, AL2 AL2 optimisée et accélérée AMIs, à compter du 26 novembre 2025. Bien que vous puissiez continuer à utiliser EKS AL2 AMIs après la date end-of-support (EOS) (26 novembre 2025), EKS ne publiera plus de nouvelles versions ou mises à jour de Kubernetes AL2 AMIs, y compris les versions mineures, les correctifs et les corrections de bogues après cette date. Nous vous recommandons de passer à Amazon Linux 2023 (AL2023) ou à AMIs Bottlerocket :

- AL2Le 023 permet une secure-by-default approche avec des politiques de sécurité préconfigurées, SELinux en mode permissif, le mode IMDSv2 uniquement activé par défaut, des temps de démarrage optimisés et une gestion des packages améliorée pour une sécurité et des performances améliorées, parfaitement adapté aux infrastructures nécessitant des personnalisations importantes, telles qu'un accès direct au niveau du système d'exploitation ou des modifications importantes des nœuds.
- Bottlerocket permet d'améliorer la sécurité, de raccourcir les temps de démarrage et de réduire la surface d'attaque pour une efficacité accrue grâce à sa conception spécialement conçue et optimisée pour les conteneurs, parfaitement adaptée aux approches natives des conteneurs avec une personnalisation minimale des nœuds.

Vous pouvez également choisir d'the section called “Constructions personnalisées”attendre la date EOS (26 novembre 2025) ou de créer une AMI personnalisée avec une instance de base Amazon Linux 2 jusqu'à la date EOS d'Amazon Linux 2 (30 juin 2026). Pour plus d'informations, rendez-vous sur AL2023 FAQs, Bottlerocket FAQs ou consultez notre the section called “Bottlerocket” documentation pour obtenir des the section called “Passez à la version AL2 023” conseils de migration détaillés.

Migration et support FAQs

Comment migrer de mon AMI AL2 vers une AMI AL2 023 ?

Nous vous recommandons de créer et de mettre en œuvre un plan de migration comprenant des tests approfondis de la charge de travail des applications et des procédures de restauration documentées, puis de suivre les step-by-step instructions de la documentation officielle relative à la mise à niveau d'Amazon Linux 2 vers Amazon Linux 2023 dans la documentation officielle d'EKS.

Puis-je créer une AL2 AMI personnalisée après la date EKS end-of-support (EOS) pour EKS Optimized AL2 AMIs ?

Bien que nous vous recommandions de passer à EKS, officiellement pris en charge et publié, optimisé AMIs pour AL2 023 ou Bottlerocket, vous pouvez créer des EKS personnalisés AL2 optimisés et accélérés AL2 jusqu'à AMIs la date AL2 AMI EOS (26 novembre 2025). Vous pouvez également créer une AMI personnalisée avec une instance de base Amazon Linux 2 jusqu'à la date Amazon Linux 2 EOS (30 juin 2026). Pour step-by-step obtenir des instructions sur la création d'une AL2 AMI personnalisée AL2 optimisée et accélérée pour EKS, consultez la section Création d'une AMI Amazon Linux personnalisée dans la documentation officielle d'EKS.

La politique de prise en charge des versions d'EKS Kubernetes s'applique-t-elle aux distributions Amazon Linux ?

Non. La date EOS pour EKS AL2 AL2 optimisée et accélérée AMIs est indépendante des délais de support standard et étendus pour les versions Kubernetes d'EKS. Vous devez migrer vers AL2 023 ou Bottlerocket même si vous utilisez le support étendu d'EKS.

Comment le passage de cgroupv1 à cgroupv2 affecte-t-il ma migration ?

La communauté Kubernetes a fait passer le `cgroupv1` support (utilisé par AL2) en mode maintenance, ce qui signifie qu'aucune nouvelle fonctionnalité ne sera ajoutée et que seules des mesures de sécurité critiques et des corrections de bogues majeures seront fournies. Pour l'adopter `cgroupv2` dans Kubernetes, vous devez garantir la compatibilité entre le système d'exploitation, le

noyau, l'environnement d'exécution du conteneur et les composants Kubernetes. Cela nécessite une distribution Linux activée `cgroupv2` par défaut, telle que AL2 023, Bottlerocket, Red Hat Enterprise Linux (RHEL) 9+, Ubuntu 22.04+ ou Debian 11+. Ces distributions sont fournies avec des versions de noyau ≥ 5.8, ce qui est la configuration minimale requise pour être `cgroupv2` prise en charge dans Kubernetes. Pour en savoir plus, consultez À propos de cgroup v2.

Que dois-je faire si j'ai besoin de Neuron dans mon AL2 AMI personnalisée ?

Vous ne pouvez pas exécuter toutes vos applications alimentées par Neuron de manière native sur une base. AL2 AMIs Pour tirer parti de AWS Neuron sur une AL2 AMI, vous devez conteneuriser vos applications à l'aide d'un conteneur compatible avec Neuron avec une distribution autre que AL2 Linux (par exemple, Ubuntu 22.04, Amazon Linux 2023, etc.), puis déployer ces conteneurs sur une AL2 AMI basée sur laquelle le pilote Neuron () est installé. `aws-neuronx-dkms`

Compatibilité et versions

Versions de Kubernetes prises en charge pour AL2 AMIs

La version 1.32 de Kubernetes est la dernière version pour laquelle Amazon EKS publiera (AL2 Amazon Linux 2). AMIs Pour les versions de Kubernetes prises en charge jusqu'à la version 1.32, EKS continuera à publier AL2 AMIs (AL2_ARM_64, _x86_64) et à accélérer (AL2_x86_64_GPU) jusqu'au 26 novembre 2025. AL2 AMIs AL2 Après cette date, EKS cessera de publier des versions AL2 optimisées et AL2 accélérées AMIs pour toutes les versions de Kubernetes. Notez que la date EOS pour EKS AL2 AL2 optimisée et accélérée AMIs est indépendante des délais de support standard et étendus pour les versions Kubernetes d'EKS.

Comparaison des pilotes NVIDIA pour AL2, AL2 023 et Bottlerocket AMIs

Branche des chauffeurs	AMI Amazon Linux 2	AMI Amazon Linux 2023	Bottlerocket AMI	End-of-Life Date
R535	Non pris en charge	Non pris en charge	Pris en charge	septembre 2027
R550	Pris en charge	Pris en charge	Non pris en charge	avril 2025
R560	Non pris en charge	Pris en charge	Non pris en charge	mars 2025

Branche des chauffeurs	AMI Amazon Linux 2	AMI Amazon Linux 2023	Bottlerocket AMI	End-of-Life Date
R570	Non pris en charge	Pris en charge	Prochainement	février 2026

Pour en savoir plus, consultez la documentation des versions de Nvidia.

Comparaison des versions de NVIDIA CUDA pour AL2, AL2 023 et Bottlerocket AMIs

Version CUDA	AL2 Support	AL2023 Support	Support Bottlerocket
10,1	Pris en charge	Non pris en charge	Non pris en charge
11.8	Pris en charge	Pris en charge	Pris en charge
12,0	Non pris en charge	Pris en charge	Pris en charge
12,5	Non pris en charge	Pris en charge	Pris en charge

Pour en savoir plus, consultez la documentation relative à la version de CUDA.

Comparaison des pilotes pris en charge et des versions du noyau Linux pour AL2, AL2 023 et Bottlerocket AMIs

Composant	AL2 Source de l'AMI	AL2023 Source de l'AMI	Source d'AMI Bottlerocket
Compatibilité du système d'exploitation de base	RHEL7/CentOS 7	Fedora/CentOS 9	N/A
CUDA Toolkit	CUDA 11,x—12,x	CUDA 12,5 et versions ultérieures	CUDA 11.x (12.5 bientôt disponible)
Pilote GPU NVIDIA	R550	R565 (R570 bientôt disponible)	R535 (R570 bientôt disponible)

Composant	AL2 Source de l'AMI	AL2023 Source de l'AMI	Source d'AMI Bottlerocket
AWS Pilote Neuron	2,19	2,19 et plus	2,20
Noyau Linux	5,10	6,1, 6,12	5.15, 6.1 (6.12 bientôt disponible)

AWS Compatibilité des neurones avec AL2 AMIs

À partir de AWS la version 2.20 de Neuron, le Neuron Runtime (`aws-neuronx-runtime-lib`) utilisé par EKS AL AMIs n'est plus compatible avec Amazon Linux 2 (). AL2 Le pilote Neuron (`aws-neuronx-dkms`) est désormais le seul package AWS Neuron compatible avec Amazon Linux 2. Cela signifie que vous ne pouvez pas exécuter vos applications alimentées par Neuron de manière native sur une AMI basée AL2 sur une AMI. Pour configurer Neuron on AL2 023 AMIs, consultez le guide de configuration AWS Neuron.

Compatibilité de Kubernetes avec AL2 AMIs

La communauté Kubernetes a déplacé le `cgroupv1` support (utilisé par AL2) en mode maintenance. Cela signifie qu'aucune nouvelle fonctionnalité ne sera ajoutée et que seules des mesures de sécurité critiques et des corrections de bogues majeures seront fournies. Toutes les fonctionnalités de Kubernetes basées sur cgroupv2, telles que MemoryQo S et l'isolation améliorée des ressources, ne sont pas disponibles sur. AL2 En outre, la version 1.32 d'Amazon EKS Kubernetes était la dernière version à être prise en charge. AL2 AMIs Pour maintenir la compatibilité avec les dernières versions de Kubernetes, nous vous recommandons de migrer vers AL2 023 ou Bottlerocket, qui sont activées par défaut. `cgroupv2`

Compatibilité des versions Linux avec AL2 AMIs

Amazon Linux 2 (AL2) est pris en charge AWS jusqu'à sa date end-of-support (EOS) du 30 juin 2026. Cependant, avec le AL2 temps, le soutien apporté par l'ensemble de la communauté Linux aux nouvelles applications et fonctionnalités est devenu plus limité. AL2 AMIs sont basés sur le noyau Linux 5.10, tandis que AL2 023 utilise le noyau Linux 6.10. Contrairement à AL2 023, AL2 bénéficie d'un soutien limité de la part de la communauté Linux au sens large. Cela signifie que de nombreux packages et outils Linux en amont doivent être rétroportés pour fonctionner avec l'ancienne version AL2 du noyau, certaines fonctionnalités modernes de Linux et certaines améliorations de sécurité ne

sont pas disponibles en raison de l'ancien noyau, de nombreux projets open source ont déconseillé ou limité le support des anciennes versions du noyau, comme la 5.10.

Packages obsolètes non inclus dans la version 023 AL2

Voici quelques-uns des packages les plus courants qui ne sont pas inclus ou qui ont changé en AL2 2023 :

- Certains packages binaires source dans Amazon Linux 2 ne sont plus disponibles dans Amazon Linux 2023
- Modifications apportées à la manière dont Amazon Linux prend en charge les différentes versions de packages (par exemple, amazon-linux-extras le système) en AL2 2023
- Les packages supplémentaires pour Enterprise Linux (EPEL) ne sont pas pris en charge dans AL2 la version 023
- Les applications 32 bits ne sont pas prises en charge dans AL2 la version 2.3
- AL2023 ne crée pas de packages mate-desktop

Pour en savoir plus, consultez Comparing AL2 et AL2 023.

Comparaison des validations FIPS entre AL2, AL2 023 et Bottlerocket

Amazon Linux 2 (AL2), Amazon Linux 2023 (AL2023) et Bottlerocket fournissent une assistance en matière de conformité aux normes fédérales de traitement de l'information (FIPS).

- AL2 est certifié selon FIPS 140-2 et AL2 023 est certifié selon FIPS 140-3. Pour activer le mode FIPS le AL2 023, installez les packages nécessaires sur votre EC2 instance Amazon et suivez les étapes de configuration en suivant les instructions de la section Activer le mode FIPS le 023. AL2 Pour en savoir plus, consultez AL2023 FAQs.
- Bottlerocket propose des variantes spécialement conçues pour FIPS, qui limitent le noyau et les composants de l'espace utilisateur à l'utilisation de modules cryptographiques soumis au programme de validation des modules cryptographiques FIPS 140-3.

Journal des modifications du pilote EKS AMI et des versions

Pour obtenir la liste complète de tous les composants de l'AMI EKS et de leurs versions, consultez les notes de mise à jour de l'AMI Amazon EKS sur GitHub.

Créez des nœuds avec Amazon Linux optimisé AMIs

Les systèmes Amazon Linux optimisés pour Amazon EKS AMIs sont basés sur Amazon Linux 2 (AL2) et Amazon Linux 2023 (AL2023). Elles sont configurées pour servir d'images de base aux nœuds Amazon EKS. Ils AMIs sont configurés pour fonctionner avec Amazon EKS et incluent les composants suivants :

- `kubelet`
- AWS Authentificateur IAM
- `containerd`

> **Note**
>
> - Vous pouvez suivre les événements liés à la sécurité ou à la confidentialité d'Amazon Linux dans le centre de sécurité Amazon Linux en choisissant l'onglet correspondant à la version de votre choix. Vous pouvez également vous abonner au flux RSS applicable. Les événements de sécurité et de confidentialité incluent une présentation du problème, les packages concernés et la manière de mettre à jour vos instances pour résoudre le problème.
> - Avant de déployer une AMI accélérée ou une AMI Arm, consultez les informations contenues dans Amazon Linux accéléré optimisé par Amazon EKS AMIs etthe section called "Arm optimisé pour Amazon EKS | Amazon Linux AMIs".
> - Les EC2 `P2` instances Amazon ne sont pas prises en charge sur Amazon EKS car elles nécessitent la version 470 ou antérieure du `NVIDIA` pilote.
> - Tous les groupes de nœuds gérés nouvellement créés dans des clusters de la version `1.30` ou d'une version plus récente utiliseront automatiquement par défaut AL2 023 comme système d'exploitation du nœud. Auparavant, les nouveaux groupes de nœuds étaient définis par défaut sur AL2. Vous pouvez continuer à l'utiliser AL2 en le choisissant comme type d'AMI lors de la création d'un nouveau groupe de nœuds.
> - Amazon EKS ne publiera plus Amazon Linux 2 (AL2) optimisé pour EKS AMIs après le 26 novembre 2025. En outre, la version Kubernetes `1.32` est la dernière version pour laquelle Amazon EKS sera publié. AL2 AMIs `1.33`À partir de la version, Amazon EKS continuera à publier les versions AL2 023 et Bottlerocket. AMIs

Amazon EKS a optimisé Amazon Linux accéléré AMIs

Les systèmes Amazon Linux accélérés optimisés pour Amazon EKS AMIs sont basés sur le standard Amazon Linux optimisé pour Amazon EKS AMIs. Elles sont configurées pour servir d'images facultatives aux nœuds Amazon EKS afin de prendre en charge les charges de travail basées sur le GPU, Inferentia et Trainium.

Outre la configuration standard optimisée de l'AMI Amazon EKS, la configuration accélérée AMIs inclut les éléments suivants :

- Pilotes NVIDIA
- `nvidia-container-toolkit`
- AWS pilote neuronal

Pour une liste des derniers composants inclus dans l'accélérateur AMIs, consultez les `amazon-eks-ami` versions du GitHub.

> **Note**
>
> - Assurez-vous de spécifier le type d'instance applicable dans votre AWS CloudFormation modèle de nœud. En utilisant l'accélérateur optimisé Amazon EKS AMIs, vous acceptez le contrat de licence utilisateur final (EULA) Cloud de NVIDIA.
> - L'accélérateur optimisé Amazon EKS AMIs était auparavant appelé Amazon EKS optimisé AMIs avec prise en charge du GPU.
> - Les versions précédentes de l'accélérateur optimisé Amazon EKS AMIs installaient le `nvidia-docker` référentiel. Le référentiel n'est plus inclus dans l'AMI Amazon EKS version `v20200529` et ultérieure.

Pour en savoir plus sur l'exécution des charges de travail sur Amazon Linux accéléré optimisé pour Amazon EKS AMIs, consultezthe section called “Configurer le GPU Linux AMIs”.

Arm optimisé pour Amazon EKS | Amazon Linux AMIs

Les instances Arm apportent des économies significatives en termes de coût et sont bien adaptées aux applications dimensionnables et basées sur Arm, telles que serveurs web, microservices

conteneurisés, flottes de cache et magasins de données distribuées. Lorsque vous ajoutez des nœuds Arm à votre cluster, passez en revue les considérations suivantes.

- Si votre cluster a été déployé avant le 17 août 2020, vous devez effectuer une mise à niveau unique des manifestes des modules complémentaires critiques du cluster. Ceci afin que Kubernetes puisse extraire l'image correcte pour chaque architecture matérielle utilisée dans votre cluster. Pour plus d'informations sur la mise à jour des modules complémentaires de clusters, consultez the section called "Étape 1 : Préparation de la mise à niveau". Si vous avez déployé votre cluster le 17 août 2020 ou après cette date, vos modules complémentaires CoreDNS et Amazon VPC `kube-proxy` CNI pour Kubernetes sont déjà compatibles avec plusieurs architectures.
- Les applications déployées sur les nœuds Arm doivent être compilées pour Arm.
- Si vous en avez DaemonSets déployé dans un cluster existant, ou si vous souhaitez les déployer sur un nouveau cluster dans lequel vous souhaitez également déployer des nœuds Arm, vérifiez que vous DaemonSet pouvez les exécuter sur toutes les architectures matérielles de votre cluster.
- Vous pouvez exécuter des groupes de nœuds Arm et des groupes de nœuds x86 dans le même cluster. Si c'est le cas, envisagez de déployer des images de conteneurs multi-architectures dans un référentiel de conteneurs tel qu'Amazon Elastic Container Registry, puis d'ajouter des sélecteurs de nœuds à vos manifestes afin que Kubernetes sache sur quelle architecture matérielle un pod peut être déployé. Pour de plus amples informations, consultez Transmission d'une image multi-architecture dans le Guide de l'utilisateur Amazon ECR et l'article de blog Présentation d'images de conteneurs multi-architectures pour Amazon ECR.

En savoir plus

Pour plus d'informations sur l'utilisation d'Amazon Linux optimisé pour Amazon EKS AMIs, consultez les sections suivantes :

- Pour utiliser Amazon Linux avec des groupes de nœuds gérés, veuillez consulter la rubrique the section called "Groupes de nœuds gérés".
- Pour lancer des nœuds Amazon Linux autogérés, veuillez consulter la rubrique the section called "Recevez les dernières nouvelles IDs".
- Pour plus d'informations sur la version, consultez the section called "Obtenir des informations sur la version".
- Pour accéder à la dernière version IDs d'Amazon Linux optimisée pour Amazon EKS AMIs, consultezthe section called "Recevez les dernières nouvelles IDs".

- Pour les scripts open source utilisés pour créer l'Amazon EKS optimisé AMIs, consultezthe section called “Constructions personnalisées”.

Mise à niveau d'Amazon Linux 2 vers Amazon Linux 2023

Les Amazon EKS optimisés AMIs sont disponibles en deux familles basées sur AL2 et AL2 023. AL2023 est un nouveau système d'exploitation basé sur Linux conçu pour fournir un environnement sécurisé, stable et performant pour vos applications cloud. Il s'agit de la nouvelle génération d'Amazon Linux d'Amazon Web Services et elle est disponible dans toutes les versions d'Amazon EKS prises en charge.

AL2023 propose plusieurs améliorations par rapport AL2 à. Pour une comparaison complète, consultez Comparing AL2 et Amazon Linux 2023 dans le guide de l'utilisateur Amazon Linux 2023. Plusieurs packages ont été ajoutés, mis à niveau et supprimés AL2. Il est vivement recommandé de tester vos applications avec AL2 023 avant de procéder à la mise à niveau. Pour obtenir la liste de toutes les modifications apportées aux packages en AL2 2023, consultez la section Modifications apportées aux packages dans Amazon Linux 2023 dans les notes de mise à jour d'Amazon Linux 2023.

Outre ces modifications, vous devez être conscient de ce qui suit :

- AL2023 introduit un nouveau processus d'initialisation des nœuds `nodeadm` qui utilise un schéma de configuration YAML. Si vous utilisez des groupes de nœuds autogérés ou une AMI avec un modèle de lancement, vous devez désormais fournir des métadonnées de cluster supplémentaires de manière explicite lors de la création d'un nouveau groupe de nœuds. Voici un exemple des paramètres minimaux requis, où `apiServerEndpointcertificateAuthority`, et le service `cidr` sont désormais requis :

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Dans AL2, les métadonnées de ces paramètres ont été découvertes à partir de l'appel d'`DescribeCluster`API Amazon EKS. Avec AL2 023, ce comportement a changé car l'appel d'API supplémentaire risque d'être limité lors de la mise à l'échelle de nœuds à grande échelle. Cette modification ne vous concerne pas si vous utilisez des groupes de nœuds gérés sans modèle de lancement ou si vous utilisez Karpenter. Pour plus d'informations sur les services `certificateAuthority` et les servicescidr, consultez `DescribeCluster`le manuel Amazon EKS API Reference.

- Pour AL2 023, modifie `nodeadm` également le format pour appliquer des paramètres à `kubelet` chaque nœud utilisant `NodeConfigSpec`. En AL2, cela a été fait avec le `--kubelet-extra-args` paramètre. Ceci est couramment utilisé pour ajouter des étiquettes et des nuances aux nœuds. L'exemple ci-dessous montre comment appliquer `maxPods` et `--node-labels` au nœud.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: test-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      maxPods: 110
    flags:
      - --node-labels=karpenter.sh/capacity-type=on-demand,karpenter.sh/nodepool=test
```

- La version CNI d'Amazon VPC `1.16.2` ou supérieure est requise pour 023. AL2
- AL2023 nécessite `IMDSv2` par défaut. `IMDSv2`présente plusieurs avantages qui contribuent à améliorer la posture de sécurité. Il utilise une méthode d'authentification orientée session qui nécessite la création d'un jeton secret dans une simple requête HTTP PUT pour démarrer la session. Le jeton d'une session peut être valide entre 1 seconde et 6 heures. Pour plus d'informations sur la manière de passer de `IMDSv1` à`IMDSv2`, consultez Transition vers l'utilisation du service de métadonnées d'instance version 2 et Profitez pleinement des avantages de votre AWS infrastructure IMDSv2 et désactivez-la IMDSv1 dans l'ensemble de celle-ci. Si vous souhaitez l'utiliser`IMDSv1`, vous pouvez toujours le faire en remplaçant manuellement les paramètres à l'aide des propriétés de lancement de l'option de métadonnées de l'instance.

> **Note**
>
> `IMDSv2`Avec AL2 023, le nombre de sauts par défaut pour les groupes de nœuds gérés peut varier :
>
> - Lorsque vous n'utilisez pas de modèle de lancement, la valeur par défaut est définie sur`1`. Cela signifie que les conteneurs n'auront pas accès aux informations d'identification du nœud via IMDS. Si vous avez besoin d'un accès par conteneur aux informations d'identification du nœud, vous pouvez toujours le faire en utilisant un modèle de EC2 lancement Amazon personnalisé.
> - Lorsque vous utilisez une AMI personnalisée dans un modèle de lancement, la valeur par défaut `HttpPutResponseHopLimit` est définie sur`2`. Vous pouvez le remplacer manuellement `HttpPutResponseHopLimit` dans le modèle de lancement.
>
> Vous pouvez également utiliser Amazon EKS Pod Identity pour fournir des informations d'identification au lieu de`IMDSv2`.

- AL2023 propose la nouvelle génération de hiérarchie unifiée des groupes de contrôle (`cgroupv2`). `cgroupv2`est utilisé pour implémenter un environnement d'exécution de conteneur, et par`systemd`. Bien que AL2 023 contienne toujours du code permettant au système de fonctionner en utilisant`cgroupv1`, cette configuration n'est ni recommandée ni prise en charge. Cette configuration sera complètement supprimée dans une future version majeure d'Amazon Linux.
- `eksctl`une version `0.176.0` ou supérieure est requise pour `eksctl` prendre en charge la version AL2 023.

Pour les groupes de nœuds gérés existants, vous pouvez effectuer une mise à niveau sur place ou une mise à blue/green niveau en fonction de la manière dont vous utilisez un modèle de lancement :

- Si vous utilisez une AMI personnalisée avec un groupe de nœuds gérés, vous pouvez effectuer une mise à niveau sur place en échangeant l'ID de l'AMI dans le modèle de lancement. Vous devez d'abord vous assurer que vos applications et toutes les données utilisateur sont transférées vers AL2 023 avant d'exécuter cette stratégie de mise à niveau.
- Si vous utilisez des groupes de nœuds gérés avec le modèle de lancement standard ou avec un modèle de lancement personnalisé qui ne précise pas l'ID de l'AMI, vous devez effectuer une mise à niveau à l'aide d'une blue/green stratégie. Une blue/green mise à niveau est généralement plus complexe et implique la création d'un tout nouveau groupe de nœuds dans lequel vous devez spécifier AL2 023 comme type d'AMI. Le nouveau groupe de nœuds devra ensuite être configuré

avec soin pour garantir que toutes les données personnalisées du groupe de AL2 nœuds sont compatibles avec le nouveau système d'exploitation. Une fois que le nouveau groupe de nœuds a été testé et validé avec vos applications, les pods peuvent être migrés de l'ancien groupe de nœuds vers le nouveau groupe de nœuds. Une fois la migration terminée, vous pouvez supprimer l'ancien groupe de nœuds.

Si vous utilisez Karpenter et que vous souhaitez utiliser AL2 023, vous devez modifier le `EC2NodeClass` `amiFamily` champ avec AL2 023. Par défaut, Drift est activé dans Karpenter. Cela signifie qu'une fois le `amiFamily` champ modifié, Karpenter mettra automatiquement à jour vos nœuds de travail avec la dernière AMI lorsqu'elle sera disponible.

Récupérez les informations de version de l'AMI Amazon Linux

Les versions d'Amazon Linux optimisées pour Amazon EKS AMIs sont définies en fonction de la version de Kubernetes et de la date de sortie de l'AMI au format suivant :

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

Chaque version de l'AMI inclut différentes versions de kubelet, du noyau Linux et de containerd. L' AMIs accélérateur inclut également différentes versions du pilote NVIDIA. Vous pouvez trouver ces informations de version dans le Journal des modifications sur GitHub.

Récupérez l'AMI Amazon Linux recommandée IDs

Lorsque vous déployez des nœuds, vous pouvez spécifier un ID pour une Amazon Machine Image (AMI) prédéfinie optimisée pour Amazon EKS. Pour récupérer un ID d'AMI correspondant à la configuration souhaitée, interrogez l'API du magasin de paramètres AWS Systems Manager. L'utilisation de cette API élimine le besoin de rechercher manuellement l'AMI optimisée pour Amazon EKS IDs. Pour de plus amples informations, veuillez consulter GetParameter. Le principal IAM que vous utilisez doit disposer de l'autorisation IAM `ssm:GetParameter` pour récupérer les métadonnées de l'AMI optimisée pour Amazon EKS.

Vous pouvez récupérer l'ID d'image de la dernière AMI Amazon Linux optimisée pour Amazon EKS recommandée à l'aide de la commande suivante, qui utilise le sous-paramètre`image_id`. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- `<kubernetes-version>`Remplacez-le par un eks/latest/userguide/platform fichier -versions.html compatible [platform-version, type="documentation "].

- *ami-type*Remplacez-le par l'une des options suivantes. Pour plus d'informations sur les types d' EC2 instances Amazon, consultez la section Types d' EC2 instances Amazon.
  - *amazon-linux-2023/x86_64/standard*À utiliser pour les instances x86 basées sur Amazon Linux 2023 (AL2023).
  - *amazon-linux-2023/arm64/standard*À utiliser pour AL2 023 instances ARM, telles que les instances basées sur AWS Graviton.
  - *amazon-linux-2023/x86_64/nvidia*À utiliser pour les dernières instances NVIDIA AL2 x86 023 approuvées.
  - *amazon-linux-2023/arm64/nvidia*À utiliser pour les dernières instances NVIDIA AL2 arm64 023 approuvées.
  - *amazon-linux-2023/x86_64/neuron*À utiliser pour les dernières instances AL2 023 AWS Neuron.
  - *amazon-linux-2*À utiliser pour les instances x86 basées sur Amazon Linux 2 (AL2).
  - *amazon-linux-2-arm64*À utiliser pour les instances AL2 ARM, telles que les instances basées sur AWS Graviton.
  - *amazon-linux-2-gpu*À utiliser pour les instances x86 basées sur l'accélération AL2 matérielle pour les charges de travail basées sur le GPU NVIDIA, Inferentia et Trainium.
- <region-code>Remplacez-le par une AWS région prise en charge par Amazon EKS pour laquelle vous souhaitez obtenir l'ID AMI.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/<kubernetes-version>/<ami-
type>/recommended/image_id \
    --region <region-code> --query "Parameter.Value" --output text
```

Voici un exemple de commande après le remplacement des espaces réservés.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.31/amazon-
linux-2023/x86_64/standard/recommended/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

L'exemple qui suit illustre un résultat.

```
ami-1234567890abcdef0
```

Création d'une AMI Amazon Linux personnalisée

> **Important**
>
> Amazon EKS ne publiera plus Amazon Linux 2 (AL2) optimisé pour EKS AMIs après le 26 novembre 2025. En outre, la version Kubernetes `1.32` est la dernière version pour laquelle Amazon EKS sera publié. AL2 AMIs `1.33`À partir de la version, Amazon EKS continuera à publier les versions AL2 023 et Bottlerocket. AMIs

Les Amazon Linux (AL) optimisés pour Amazon EKS AMIs sont basés sur AL2 et AL2 023, spécifiquement pour être utilisés comme nœuds dans les clusters Amazon EKS. Amazon EKS fournit des scripts de génération open source dans le référentiel de spécifications de construction AMI Amazon EKS que vous pouvez utiliser de la manière suivante :* Consultez les configurations`kubelet`, le moteur d'exécution et l'authentificateur AWS IAM pour Kubernetes. * Créez votre propre AMI basée sur AL à partir de zéro.

Ce dépôt contient le script bootstrap spécialisé et le script nodeadm qui s'exécutent au démarrage. Ces scripts configurent les données de certificat de votre instance, le point de terminaison du plan de contrôle, le nom du cluster, etc. Les scripts sont considérés comme la source de vérité pour les builds d'AMI optimisés pour Amazon EKS. Vous pouvez donc suivre le GitHub référentiel pour suivre les modifications apportées à notre AMIs.

Prérequis

- Installation de la AWS CLI
- Installez HashiCorp Packer v1.9.4+
- Installez GNU Make

Quickstart

Cette section présente les commandes permettant de créer une AMI personnalisée dans votre AWS compte. Pour en savoir plus sur les configurations disponibles pour personnaliser votre AMI, consultez les variables de modèle sur la page Amazon Linux 2023.

Étape 1. Configurez votre environnement

Clonez ou dupliquez le référentiel officiel d'AMI Amazon EKS. Par exemple :

```
git clone https://github.com/awslabs/amazon-eks-ami.git
cd amazon-eks-ami
```

Vérifiez que Packer est installé :

```
packer --version
```

Étape 2. Pour créer une AMI personnalisée

Vous trouverez ci-dessous des exemples de commandes pour différentes commandes personnalisées AMIs.

AL2 AMI NVIDIA de base :

```
make k8s=1.31 os_distro=al2 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI NVIDIA AL2 023 de base :

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI Neuron AL2 023 conforme aux STIG :

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=neuron \
  enable_fips=true \
  source_ami_id=ami-0abcd1234efgh5678 \
  kms_key_id=alias/aws-stig
```

Après avoir exécuté ces commandes, Packer effectuera les opérations suivantes :* Lancer une EC2 instance Amazon temporaire. * Installez les composants, les pilotes et les configurations de Kubernetes. * Créez l'AMI dans votre AWS compte.

Étape 3. Afficher les valeurs par défaut

Pour afficher les valeurs par défaut et les options supplémentaires, exécutez la commande suivante :

```
make help
```

Créez des nœuds avec Bottlerocket optimisé AMIs

Bottlerocket est une distribution Linux open source sponsorisée et soutenue par. AWS Bottlerocket est spécialement conçu pour héberger des charges de travail de conteneurs. Avec Bottlerocket, vous pouvez améliorer la disponibilité des déploiements conteneurisés et réduire les coûts opérationnels en automatisant les mises à jour de votre infrastructure de conteneurs. Bottlerocket inclut uniquement les logiciels essentiels pour exécuter les conteneurs, ce qui améliore l'utilisation des ressources, réduit les menaces de sécurité et réduit les frais de gestion. L'AMI Bottlerocket `containerd` inclut`kubelet`, AWS et l'authentificateur IAM. Outre les groupes de nœuds gérés et les nœuds autogérés, Bottlerocket est également pris en charge par Karpenter.

Avantages

L'utilisation de Bottlerocket avec votre cluster Amazon EKS présente les bénéfices suivants :

- Disponibilité accrue, coûts d'exploitation réduits et complexité de gestion réduite : Bottlerocket occupe moins de ressources, a des temps de démarrage plus courts et est moins vulnérable aux menaces de sécurité que les autres distributions Linux. L'encombrement réduit de Bottlerocket permet de réduire les coûts en utilisant moins de ressources de stockage, de calcul et de réseau.
- Sécurité améliorée grâce aux mises à jour automatiques du système d'exploitation – les mises à jour de Bottlerocket sont appliquées en une seule unité et peuvent être annulées si nécessaire. Cela élimine le risque de mises à jour corrompues ou échouées qui peuvent laisser le système dans un état inutilisable. Avec Bottlerocket, les mises à jour de sécurité peuvent être appliquées automatiquement dès qu'elles sont disponibles de manière peu perturbatrice et être annulées en cas de défaillance.
- Support premium : à AWS condition que les versions de Bottlerocket sur Amazon EC2 soient couvertes par les mêmes plans de AWS support qui couvrent également AWS des services tels qu'Amazon, EC2 Amazon EKS et Amazon ECR.

Considérations

Tenez compte des points suivants lorsque vous utilisez Bottlerocket pour votre type d'AMI :

- Bottlerocket prend en charge les EC2 instances Amazon avec `x86_64` et les processeurs. `arm64` L'AMI Bottlerocket n'est pas recommandée pour les instances EC2 Amazon dotées d'une puce Inferentia.
- Les images Bottlerocket n'incluent ni serveur SSH ni shell. Vous pouvez utiliser des méthodes out-of-band d'accès pour autoriser le SSH. Ces approches activent le conteneur d'administration et transmettent certaines étapes de configuration de l'action d'amorçage avec des données utilisateur. Pour plus d'informations, consultez les sections suivantes de Bottlerocket OS sur : GitHub
  - Exploration
  - Conteneur d'administration
  - Paramètres Kubernetes
- Bottlerocket utilise différents types de contenants :
  - Par défaut, est activé un conteneur de contrôle. Ce conteneur exécute l'agent AWS Systems Manager que vous pouvez utiliser pour exécuter des commandes ou démarrer des sessions shell sur des instances Amazon EC2 Bottlerocket. Pour plus d'informations, consultez la section Configuration du gestionnaire de session dans le guide de l'utilisateur de AWS Systems Manager.
  - Un conteneur d'administration est activé si une clé SSH est fournie lors de la création du groupe de nœuds. Nous recommandons d'utiliser le conteneur d'administration uniquement pour les scénarios de développement et de test. Nous ne recommandons pas de l'utiliser pour les environnements de production. Pour plus d'informations, consultez Conteneur d'administration sur GitHub.

En savoir plus

Pour plus d'informations sur l'utilisation de Bottlerocket optimisé pour Amazon EKS AMIs, consultez les sections suivantes :

- Pour plus de détails sur Bottlerocket, consultez la documentation de Bottlerocket.
- Pour les ressources d'informations sur les versions, voirthe section called “Obtenir des informations sur la version”.
- Pour utiliser Bottlerocket avec des groupes de nœuds gérés, consultez. the section called “Groupes de nœuds gérés”
- Pour lancer des nœuds Bottlerocket autogérés, consultez. the section called “Bottlerocket”
- Pour récupérer la dernière version IDs du Bottlerocket AMIs optimisé pour Amazon EKS, consultez. the section called “Recevez les dernières nouvelles IDs”

- Pour plus de détails sur la prise en charge en matière de conformité, veuillez consulter la rubrique the section called "Assistance en matière de conformité".

Récupérez les informations de version de l'AMI Bottlerocket

Chaque version de l'AMI Bottlerocket inclut différentes versions de kubelet, du noyau Bottlerocket et de containerd. Les variantes d'AMI accélérées incluent également différentes versions du pilote NVIDIA. Vous trouverez les informations relatives à cette version dans la rubrique OS de la documentation de Bottlerocket. À partir de cette page, accédez à la sous-rubrique Informations de version applicable.

La documentation de Bottlerocket peut parfois être en retard par rapport aux versions disponibles sur. GitHub Vous trouverez une liste des modifications apportées aux dernières versions dans les versions sur GitHub.

Récupérez l'AMI Bottlerocket recommandée IDs

Lorsque vous déployez des nœuds, vous pouvez spécifier un ID pour une Amazon Machine Image (AMI) prédéfinie optimisée pour Amazon EKS. Pour récupérer un ID d'AMI correspondant à la configuration souhaitée, interrogez l'API du magasin de paramètres AWS Systems Manager. L'utilisation de cette API élimine le besoin de rechercher manuellement l'AMI optimisée pour Amazon EKS IDs. Pour de plus amples informations, veuillez consulter GetParameter. Le principal IAM que vous utilisez doit disposer de l'autorisation IAM `ssm:GetParameter` pour récupérer les métadonnées de l'AMI optimisée pour Amazon EKS.

Vous pouvez récupérer l'ID d'image de la dernière AMI Bottlerocket optimisée pour Amazon EKS recommandée à l'aide de la commande AWS CLI suivante, qui utilise le sous-paramètre. `image_id` Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- *`kubernetes-version`*Remplacez-le par un eks/latest/userguide/platform fichier -versions.html compatible [platform-version, type="documentation "].
- *`-flavor`*Remplacez-le par l'une des options suivantes.
  - Supprimer *`-flavor`* pour les variantes sans GPU.
  - À utiliser *`-nvidia`* pour les variantes compatibles avec le GPU.
  - À utiliser *`-fips`* pour les variantes compatibles FIPS.
- *`architecture`*Remplacez-le par l'une des options suivantes.

- À utiliser *x86_64* pour les instances x86 basées.
- À utiliser *arm64* pour les instances ARM.

- *region-code*Remplacez-le par une AWS région prise en charge par Amazon EKS pour laquelle vous souhaitez obtenir l'ID AMI.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-
flavor/architecture/latest/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Voici un exemple de commande après le remplacement des espaces réservés.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.31/x86_64/latest/
image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

L'exemple qui suit illustre un résultat.

```
ami-1234567890abcdef0
```

Répondez aux exigences de conformité avec Bottlerocket

Bottlerocket respecte les recommandations définies par différents organismes :

- Il existe un benchmark CIS défini pour Bottlerocket. Dans une configuration par défaut, l'image Bottlerocket possède la plupart des contrôles requis par le profil de configuration CIS de niveau 1. Vous pouvez implémenter les contrôles requis pour un profil de configuration CIS de niveau 2. Pour plus d'informations, consultez la section Validation de l'AMI Bottlerocket optimisée pour Amazon EKS par rapport au CIS Benchmark sur le blog. AWS
- L'ensemble de fonctionnalités optimisées et la surface d'attaque réduite signifient que les instances de Bottlerocket nécessitent moins de configuration pour répondre aux exigences de la norme PCI DSS. Le référentiel CIS pour Bottlerocket est une excellente ressource pour renforcer les conseils et répond à vos exigences en matière de normes de configuration sécurisées conformément à l'exigence PCI DSS 2.2. Vous pouvez également utiliser Fluent Bit pour répondre à vos exigences en matière de journalisation des audits au niveau du système d'exploitation conformément à l'exigence 10.2 de la norme PCI DSS. AWS publie régulièrement de nouvelles instances

Bottlerocket (corrigées) pour vous aider à répondre aux exigences de la norme PCI DSS 6.2 (pour v3.2.1) et à l'exigence 6.3.3 (pour v4.0).

- Bottlerocket est une fonctionnalité éligible à la loi HIPAA autorisée à être utilisée avec des charges de travail réglementées à la fois pour Amazon et Amazon EKS. EC2 Pour plus d'informations, consultez la référence des services éligibles à la loi HIPAA.
- AMIs Il existe des modèles Bottlerocket préconfigurés pour utiliser des modules cryptographiques validés par la norme FIPS 140-3. Cela inclut le module cryptographique de l'API Kernel Crypto d'Amazon Linux 2023 et le module cryptographique AWS-LC. Pour de plus amples informations, veuillez consulter the section called "Bottlerocket FIPS AMIs".

Préparez vos nœuds de travail à la norme FIPS avec Bottlerocket FIPS AMIs

La publication 140-3 de la Federal Information Processing Standard (FIPS) est une norme gouvernementale des États-Unis d'Amérique et du Canada qui spécifie les exigences de sécurité pour les modules cryptographiques qui protègent les informations sensibles. Bottlerocket facilite l'adhésion à la norme FIPS en proposant un AMIs noyau FIPS.

Ils AMIs sont préconfigurés pour utiliser des modules cryptographiques validés par la norme FIPS 140-3. Cela inclut le module cryptographique de l'API Kernel Crypto d'Amazon Linux 2023 et le module cryptographique AWS-LC.

L'utilisation de Bottlerocket FIPS AMIs rend vos nœuds de travail « prêts pour la norme FIPS », mais pas automatiquement « conformes à la norme FIPS ». Pour plus d'informations, consultez la norme fédérale de traitement de l'information (FIPS) 140-3.

Considérations

- Si votre cluster utilise des sous-réseaux isolés, le point de terminaison Amazon ECR FIPS risque de ne pas être accessible. Cela peut entraîner l'échec du bootstrap du nœud. Assurez-vous que la configuration de votre réseau autorise l'accès aux points de terminaison FIPS nécessaires. Pour plus d'informations, consultez la section Accès à une ressource via un point de terminaison VPC de ressource dans le AWS PrivateLink Guide.
- Si votre cluster utilise un sous-réseau avec PrivateLink, les extractions d'images échoueront car les points de terminaison FIPS Amazon ECR ne sont pas disponibles via. PrivateLink

Création d'un groupe de nœuds géré avec une AMI Bottlerocket FIPS

L'AMI FIPS Bottlerocket est disponible en deux variantes pour prendre en charge vos charges de travail :

- `BOTTLEROCKET_x86_64_FIPS`
- `BOTTLEROCKET_ARM_64_FIPS`

Pour créer un groupe de nœuds gérés avec une AMI Bottlerocket FIPS, choisissez le type d'AMI applicable lors du processus de création. Pour de plus amples informations, veuillez consulter the section called "Création".

Pour plus d'informations sur la sélection des variantes compatibles FIPS, consultez. the section called "Recevez les dernières nouvelles IDs"

Désactiver le point de terminaison FIPS pour les régions non prises en charge AWS

Les systèmes FIPS Bottlerocket AMIs sont pris en charge directement aux États-Unis d'Amérique, y compris dans les régions (américaines). AWS GovCloud Pour AWS les régions où AMIs ils sont disponibles mais ne sont pas directement pris en charge, vous pouvez toujours les utiliser AMIs en créant un groupe de nœuds gérés avec un modèle de lancement.

L'AMI Bottlerocket FIPS repose sur le point de terminaison Amazon ECR FIPS lors du démarrage, qui n'est généralement pas disponible en dehors des États-Unis d'Amérique. Pour utiliser l'AMI pour son noyau FIPS dans les AWS régions où le point de terminaison Amazon ECR FIPS n'est pas disponible, procédez comme suit pour désactiver le point de terminaison FIPS :

1. Créez un nouveau fichier de configuration avec le contenu suivant ou incorporez-le dans votre fichier de configuration existant.

```
[default]
use_fips_endpoint=false
```

1. Codez le contenu du fichier au format Base64.
2. Dans votre modèle de lancement`UserData`, ajoutez la chaîne codée suivante au format TOML :

```
[settings.aws]
```

```
config = "<your-base64-encoded-string>"
```

Pour les autres paramètres, consultez la description des paramètres de Bottlerocket sur. GitHub

Voici un exemple `UserData` de modèle de lancement :

```
[settings]
motd = "Hello from eksctl!"
[settings.aws]
config = "W2RlZmF1bHRdCnVzZV9maXBzX2VuZHBvaW50PWZhbHNlCg==" # Base64-encoded string.
[settings.kubernetes]
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
cluster-name = "<cluster-name>"
...<other-settings>
```

Pour plus d'informations sur la création d'un modèle de lancement à partir de données utilisateur, consultezthe section called "Modèles de lancement".

Créez des nœuds avec Ubuntu Linux optimisé AMIs

Canonical s'est associé à Amazon EKS pour créer un nœud AMIs que vous pouvez utiliser dans vos clusters.

Canonical fournit une image du système d' built-for-purposeexploitation Kubernetes Node. Cette image Ubuntu réduite est optimisée pour Amazon EKS et inclut le AWS noyau personnalisé développé conjointement avec AWS. Pour plus d'informations, consultez Ubuntu sur Amazon Elastic Kubernetes Service (EKS) et. the section called "Ubuntu Linux" Pour plus d'informations sur le support, consultez la section Logiciels tiers du Support AWS Premium FAQs.

Créez des nœuds avec Windows optimisé AMIs

Les versions optimisées pour Windows Amazon EKS AMIs sont basées sur Windows Server 2019 et Windows Server 2022. Elles sont configurées pour servir d'image de base pour les nœuds Amazon EKS. Par défaut, ils AMIs incluent les composants suivants :

- kubelet
- kube-proxy
- AWS Authentificateur IAM pour Kubernetes
- proxy csi

- containerd

> **Note**
>
> Vous pouvez suivre les événements de sécurité ou de confidentialité pour Windows Server à l'aide du Guide de mise à jour de sécurité Microsoft.

Amazon EKS propose AMIs des offres optimisées pour les conteneurs Windows dans les variantes suivantes :

- AMI Windows Server 2019 Core optimisée pour Amazon EKS
- AMI Windows Server 2019 Full optimisée pour Amazon EKS
- AMI principale pour Windows Server 2022 optimisée pour Amazon EKS
- AMI complète pour Windows Server 2022 optimisée pour Amazon EKS

> **Important**
>
> - L'AMI Windows Server 20H2 Core optimisée pour Amazon EKS est obsolète. Aucune nouvelle version de cette AMI ne sera publiée.
> - Pour garantir que vous disposez des dernières mises à jour de sécurité par défaut, Amazon EKS gère Windows optimisé au cours AMIs des 4 derniers mois. Chaque nouvelle AMI sera disponible pendant 4 mois à compter de la date de publication initiale. Après cette période, AMIs les anciens fichiers deviennent privés et ne sont plus accessibles. Nous vous encourageons à utiliser la dernière version AMIs afin d'éviter les failles de sécurité et de perdre l'accès aux versions plus anciennes AMIs qui ont atteint la fin de leur durée de vie prise en charge. Bien que nous ne puissions pas garantir que nous puissions fournir un accès à AMIs ce qui a été rendu privé, vous pouvez demander l'accès en déposant un ticket auprès du AWS Support.

Calendrier des versions

Le tableau suivant répertorie les dates de publication et de fin de support pour les versions Windows sur Amazon EKS. Si une date de fin est vide, c'est parce que la version est toujours prise en charge.

Version Windows	Version d'Amazon EKS	Fin de support pour Amazon EKS
Windows Server 2022 Core	17 octobre 2022	
Windows Server 2022 Full	17 octobre 2022	
Windows Server 20H2 Core	8/12/2021	09/08/2022
Windows Server 2004 Core	19/08/2020	14/12/2021
Windows Server 2019 Core	07/10/2019	
Windows Server 2019 Full	07/10/2019	
Windows Server 1909 Core	07/10/2019	8/12/2020

Paramètres de configuration du script d'amorçage

Lorsque vous créez un nœud Windows, celui-ci contient un script qui permet de configurer différents paramètres. En fonction de votre configuration, ce script peut être trouvé sur le nœud à un emplacement similaire à :`C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. Vous pouvez indiquer des valeurs de paramètres personnalisées en les spécifiant comme arguments dans le script d'amorçage. Par exemple, vous pouvez mettre à jour les données utilisateur dans le modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Données EC2 utilisateur d'Amazon".

Le script comprend les paramètres de ligne de commande suivants :

- `-EKSClusterName` : spécifie le nom du cluster Amazon EKS que ce composant master doit rejoindre.
- `-KubeletExtraArgs` : spécifie des arguments supplémentaires pour `kubelet` (facultatif).
- `-KubeProxyExtraArgs` : spécifie des arguments supplémentaires pour `kube-proxy` (facultatif).
- `-APIServerEndpoint` : spécifie le point de terminaison du serveur d'API de cluster Amazon EKS (facultatif). Uniquement valable lorsqu'il est utilisé avec `-Base64ClusterCA`. Évite d'appeler `Get-EKSCluster`.

- `-Base64ClusterCA` : spécifie le contenu de l'autorité de certification du cluster codée en base64 (facultatif). Uniquement valable lorsqu'il est utilisé avec `-APIServerEndpoint`. Évite d'appeler `Get-EKSCluster`.
- `-DNSClusterIP` : remplace l'adresse IP à utiliser pour les requêtes DNS au sein du cluster (facultatif). La valeur par défaut est `10.100.0.10` ou `172.20.0.10` en fonction de l'adresse IP de l'interface principale.
- `-ServiceCIDR`— Remplace la plage d'adresses IP du service Kubernetes à partir de laquelle les services de cluster sont adressés. La valeur par défaut est `172.20.0.0/16` ou `10.100.0.0/16` en fonction de l'adresse IP de l'interface principale.
- `-ExcludedSnatCIDRs`— Liste des adresses `IPv4` CIDRs à exclure de la traduction des adresses réseau source (SNAT). Cela signifie que l'adresse IP privée du pod qui est adressable par VPC ne serait pas traduite en adresse IP de l'`IPv4`adresse principale de l'instance ENI pour le trafic sortant. Par défaut, le `IPv4` CIDR du VPC pour le nœud Windows Amazon EKS est ajouté. La spécification CIDRs de ce paramètre exclut également le paramètre spécifié CIDRs. Pour de plus amples informations, veuillez consulter the section called "Trafic sortant".

En plus des paramètres de la ligne de commande, vous pouvez également spécifier certains paramètres de la variable d'environnement. Lorsqu'un paramètre de ligne de commande est spécifié, il est prioritaire sur la variable d'environnement correspondante. La ou les variables d'environnement doivent être définies au niveau de la machine (ou du système), car le script d'amorçage ne lira que les variables au niveau de la machine.

Le script prend en compte les variables d'environnement suivantes :

- `SERVICE_IPV4_CIDR`— Reportez-vous au paramètre de la ligne de commande `ServiceCIDR` pour la définition.
- `EXCLUDED_SNAT_CIDRS`— Il doit s'agir d'une chaîne de caractères séparée par des virgules. Reportez-vous au paramètre de la ligne de commande `ExcludedSnatCIDRs` pour la définition.

Prise en charge de l'authentification GMSA

Les Amazon EKS Windows Pods permettent différents types d'authentification par compte de service géré de groupe (GMSA).

- Amazon EKS prend en charge les identités de domaine Active Directory pour l'authentification. Pour plus d'informations sur les GMSA joints à un domaine, consultez Windows Authentication on Amazon EKS Windowspods sur le blog. AWS
- Amazon EKS propose un plug-in qui permet aux nœuds non-domain-joined Windows de récupérer les informations d'identification GMSA à l'aide d'une identité utilisateur portable. Pour plus d'informations sur les GMSA sans domaine, consultez Domainless Windows Authentication for Amazon EKS Windowspods sur le blog. AWS

Images de conteneur en cache

Certaines images de conteneur AMIs optimisées pour Amazon EKS pour Windows sont mises en cache pour l'`containerd`exécution. Les images de conteneur sont mises en cache lors de la création personnalisée à AMIs l'aide de composants de génération gérés par Amazon. Pour de plus amples informations, veuillez consulter the section called "Utilisation du composant de génération géré par Amazon".

Les images de conteneur mises en cache suivantes sont destinées à l'exécution `containerd` :

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

En savoir plus

Pour plus d'informations sur l'utilisation de Windows optimisé pour Amazon EKS AMIs, consultez les sections suivantes :

- Pour en savoir plus sur l'exécution des charges de travail sur un système Windows accéléré optimisé pour Amazon EKS AMIs, consultezthe section called "Configurer le GPU Windows AMIs".
- Pour utiliser Windows avec des groupes de nœuds gérés, consultezthe section called "Groupes de nœuds gérés".
- Pour lancer des nœuds Windows autogérés, consultezthe section called "Windows".
- Pour plus d'informations sur la version, consultez the section called "Obtenir des informations sur la version".
- Pour accéder à la dernière version IDs de Windows optimisée pour Amazon EKS AMIs, consultezthe section called "Recevez les dernières nouvelles IDs".

- Pour utiliser Amazon EC2 Image Builder afin de créer des fenêtres personnalisées optimisées pour Amazon EKS AMIs, consultezthe section called "Constructions personnalisées".
- Pour connaître les meilleures pratiques, consultez la section Gestion optimisée des AMI Windows Amazon EKS dans le guide des meilleures pratiques d'EKS.

Créez des nœuds Windows Server 2022 autogérés avec **`eksctl`**

Vous pouvez utiliser ce qui suit `test-windows-2022.yaml` comme référence pour créer des nœuds Windows Server 2022 autogérés. Remplacez chaque *`example value`* par vos propres valeurs.

> **Note**
>
> Vous devez utiliser `eksctl` la version 0.116.0 ou ultérieure pour exécuter des nœuds Windows Server 2022 autogérés.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.33'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

Les groupes de nœuds peuvent ensuite être créés à l'aide de la commande suivante.

```
eksctl create cluster -f test-windows-2022.yaml
```

Récupérer les informations de version de l'AMI Windows

Cette rubrique répertorie les versions de Windows optimisé pour Amazon EKS AMIs et les versions correspondantes de kubelet, containerd etcsi-proxy.

Les métadonnées d'AMI optimisées pour Amazon EKS, notamment l'ID d'AMI, peuvent être extraites par programmation pour chaque variante. Pour de plus amples informations, veuillez consulter the section called "Recevez les dernières nouvelles IDs".

AMIs sont versionnés en fonction de la version de Kubernetes et de la date de sortie de l'AMI au format suivant :

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Les groupes de nœuds gérés par Amazon EKS prennent en charge les versions de novembre 2022 et ultérieures de Windows AMIs.

Pour recevoir des notifications concernant toutes les modifications apportées au fichier source sur cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante avec un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-
ami-versions-windows.adoc.atom
```

AMI principale optimisée pour Windows Server 2022 pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI Windows Server 2022 Core optimisée pour Amazon EKS.

Example

Version 1.33 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.33-2025.07.16`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.06.13`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.05.17`	`1.33.1`	`1.7.27`	`1.2.1`	

Version 1.32 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.32-2025.07.16`	`1.32.5`	`1.7.27`	`1.2.1`	
`1.32-2025.06.13`	`1.32.5`	`1.7.27`	`1.2.1`	Mise à niveau `containerd` vers `1.7.27`.
`1.32-2025.05.17`	`1.32.5`	`1.7.20`	`1.2.1`	
`1.32-2025.04.14`	`1.32.1`	`1.7.20`	`1.1.3`	
`1.32-2025.03.14`	`1.32.1`	`1.7.20`	`1.1.3`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Version 1.31 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
				`CVE-2024-9042` .
`1.31-2024.12.13`	`1.31.3`	`1.7.20`	`1.1.3`	
`1.31-2024.11.12`	`1.31.1`	`1.7.20`	`1.1.3`	
`1.31-2024.10.08`	`1.31.1`	`1.7.20`	`1.1.3`	
`1.31-2024.10.01`	`1.31.1`	`1.7.20`	`1.1.3`	
`1.31-2024.09.10`	`1.31.0`	`1.7.20`	`1.1.3`	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.30-2025.07.16`	`1.30.13`	`1.7.27`	`1.2.1`	
`1.30-2025.06.13`	`1.30.13`	`1.7.27`	`1.2.1`	Mise à niveau `containerd` vers `1.7.27`.
`1.30-2025.05.17`	`1.30.13`	`1.7.20`	`1.2.1`	
`1.30-2025.04.14`	`1.30.9`	`1.7.20`	`1.1.3`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour `CVE-2024-5321` .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau `containerd` vers `1.7.14`.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11. Mise à niveau kubelet vers 1.29.3.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.29-2024.04.09`	`1.29.0`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilisation `golang` `1.22.1`.
`1.29-2024.03.12`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Correction d'un bug à cause duquel l'image de pause était supprimée par erreur par le processus de collecte `kubelet` des déchets.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	Windows Update KB5034439 autonome exclu sur Windows Server 2022 Core. AMIs La base de connaissances s'applique uniquement aux installations Windows avec une partition WinRE distincte, qui ne sont incluses dans aucun de nos systèmes Windows AMIs optimisés pour Amazon EKS.

Version 1.28 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.28-2024.08.13`	`1.28.11`	`1.7.14`	`1.1.3`	
`1.28-2024.07.10`	`1.28.11`	`1.7.11`	`1.1.2`	Comprend des correctifs pour `CVE-2024-5321` .
`1.28-2024.06.17`	`1.28.8`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.28-2024.05.14`	`1.28.8`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.28.8`.
`1.28-2024.04.09`	`1.28.5`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang` `1.22.1`.
`1.28-2024.03.12`	`1.28.5`	`1.6.18`	`1.1.2`	
`1.28-2024.02.13`	`1.28.5`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Windows Update KB5034439 autonome exclu sur Windows Server 2022 Core. AMIs La base de connaissances s'applique uniquement aux installations Windows avec une partition WinRE distincte, qui ne sont incluses dans aucun de nos systèmes Windows AMIs optimisés pour Amazon EKS.
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Version 1.27 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2024.08.13	1.27.15	1.7.14	1.1.3	
1.27-2024.07.10	1.27.15	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.27-2024.06.17	1.27.12	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11.
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	Mise à niveau containerd vers 1.6.28. Mise à niveau kubelet vers 1.27.12.
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	Mise à niveau containerd vers 1.6.25. CNI reconstruit et csi-proxy utilisation golang 1.22.1.
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2024.01.11	1.27.9	1.6.18	1.1.2	Windows Update KB5034439 autonome exclu sur Windows Server 2022 Core. AMIs La base de connaissances s'applique uniquement aux installations Windows avec une partition WinRE distincte, qui ne sont incluses dans aucun de nos systèmes Windows AMIs optimisés pour Amazon EKS.
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2023.10.19`	`1.27.6`	`1.6.18`	`1.1.2`	Mise à niveau `containerd` vers `1.6.18`. Ajout de nouvelles variables d'environnement pour le script d'amorçage (`SERVICE_IPV4_CIDR` et `EXCLUDED_SNAT_CIDRS`).
`1.27-2023-09.27`	`1.27.6`	`1.6.6`	`1.1.2`	Correction d'une recommandation de sécurité dans `kubelet`.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2023.06.20`	`1.27.1`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.27-2023.06.14`	`1.27.1`	`1.6.6`	`1.1.1`	Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.
`1.27-2023.06.06`	`1.27.1`	`1.6.6`	`1.1.1`	Correction du problème #2042 `containers-roadmap` qui empêchait les nœuds d'extraire des images Amazon ECR privées.
`1.27-2023.05.17`	`1.27.1`	`1.6.6`	`1.1.1`	

Version 1.26 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2025.05.17`	`1.26.15`	`1.7.20`	`1.2.1`	
`1.26-2025.04.14`	`1.26.15`	`1.7.20`	`1.1.3`	
`1.26-2025.03.14`	`1.26.15`	`1.7.20`	`1.1.3`	Mise à niveau `containerd` vers `1.7.20`.
`1.26-2025.02.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2025.01.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.12.11`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.11.12`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.10.08`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.09.10`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.08.13`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.07.10`	`1.26.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
				`CVE-2024-5321` .
`1.26-2024.06.17`	`1.26.15`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.26-2024.05.14`	`1.26.15`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.26.15`.
`1.26-2024.04.09`	`1.26.12`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang` `1.22.1`.
`1.26-2024.03.12`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.02.13`	`1.26.12`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2024.01.11	1.26.12	1.6.18	1.1.2	Windows Update KB5034439 autonome exclu sur Windows Server 2022 Core. AMIs La base de connaissances s'applique uniquement aux installations Windows avec une partition WinRE distincte, qui ne sont incluses dans aucun de nos systèmes Windows AMIs optimisés pour Amazon EKS.
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.10.19`	`1.26.9`	`1.6.18`	`1.1.2`	Mise à niveau `containerd` vers `1.6.18`. Mise à niveau `kubelet` vers `1.26.9`. Ajout de nouvelles variables d'environnement pour le script d'amorçage (`SERVICE_IPV4_CIDR` et `EXCLUDED_SNAT_CIDRS`).
`1.26-2023.09.12`	`1.26.7`	`1.6.6`	`1.1.2`	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Mise à niveau de Kubernetes vers. 1.26.4 Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	Correction d'un bogue provoquant le problème de connectivité réseau #1126 sur les pods après le redémarrage du nœud. Ajout d'un nouveau paramètre de configuration du script d'amorçage (ExcludedSnatCIDRs).
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	Ajout d'un mécanisme de récupération pour kubelet et kube-proxy en cas de panne de service.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

AMI complète optimisée pour Windows Server 2022 pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI complète Windows Server 2022 optimisée pour Amazon EKS.

Example

Version 1.33 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.33-2025.07.16`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.06.13`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.05.17`	`1.33.1`	`1.7.27`	`1.2.1`	

Version 1.32 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.32-2025.07.16`	`1.32.5`	`1.7.27`	`1.2.1`	
`1.32-2025.06.13`	`1.32.5`	`1.7.27`	`1.2.1`	Mis à niveau `containrd` vers `1.7.27`
`1.32-2025.05.17`	`1.32.5`	`1.7.20`	`1.2.1`	
`1.32-2025.04.14`	`1.32.1`	`1.7.20`	`1.1.3`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.01	1.32.0	1.7.20	1.1.3	

Version 1.31 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `contianerd` vers `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau containerd vers 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau containerd vers 1.7.20.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.29-2024.05.15`	`1.29.3`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`. Mise à niveau `kubelet` vers `1.29.3`.
`1.29-2024.04.09`	`1.29.0`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.29-2024.03.12`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Correction d'un bug à cause duquel l'image de pause était supprimée par erreur par le processus de collecte `kubelet` des déchets.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Version 1.28 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Mise à niveau containerd vers 1.6.28. Mise à niveau kubelet vers 1.28.8.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.28-2024.04.09`	`1.28.5`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.28-2024.03.12`	`1.28.5`	`1.6.18`	`1.1.2`	
`1.28-2024.02.13`	`1.28.5`	`1.6.18`	`1.1.2`	
`1.28-2024.01.09`	`1.28.5`	`1.6.18`	`1.1.2`	
`1.28-2023.12.12`	`1.28.3`	`1.6.18`	`1.1.2`	
`1.28-2023.11.14`	`1.28.3`	`1.6.18`	`1.1.2`	Comprend des correctifs pour `CVE-2023-5528` .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.28-2023.10.19`	`1.28.2`	`1.6.18`	`1.1.2`	Mise à niveau `containerd` vers `1.6.18`. Ajout de nouvelles variables d'environnement pour le script d'amorçage (`SERVICE_IPV4_CIDR` et `EXCLUDED_SNAT_CIDRS`).
`1.28-2023-09.27`	`1.28.2`	`1.6.6`	`1.1.2`	Correction d'une recommandation de sécurité dans `kubelet`.
`1.28-2023.09.12`	`1.28.1`	`1.6.6`	`1.1.2`	

Version 1.27 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2025.07.16`	`1.27.16`	`1.7.27`	`1.2.1`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2024.08.13`	`1.27.15`	`1.7.14`	`1.1.3`	
`1.27-2024.07.10`	`1.27.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour `CVE-2024-5321` .
`1.27-2024.06.17`	`1.27.12`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.27-2024.05.14`	`1.27.12`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.27.12`.
`1.27-2024.04.09`	`1.27.9`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang` `1.22.1`.
`1.27-2024.03.12`	`1.27.9`	`1.6.18`	`1.1.2`	
`1.27-2024.02.13`	`1.27.9`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2023.06.20`	`1.27.1`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.27-2023.06.14`	`1.27.1`	`1.6.6`	`1.1.1`	Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.
`1.27-2023.06.06`	`1.27.1`	`1.6.6`	`1.1.1`	Correction du problème #2042 `containers-roadmap` qui empêchait les nœuds d'extraire des images Amazon ECR privées.
`1.27-2023.05.18`	`1.27.1`	`1.6.6`	`1.1.1`	

Version 1.26 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2025.05.17`	`1.26.15`	`1.7.20`	`1.2.1`	
`1.26-2025.04.14`	`1.26.15`	`1.7.20`	`1.1.3`	
`1.26-2025.03.14`	`1.26.15`	`1.7.20`	`1.1.3`	Mise à niveau `containerd` vers `1.7.20`.
`1.26-2025.02.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2025.01.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.12.11`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.11.12`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.10.08`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.09.10`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.08.13`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.07.10`	`1.26.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
				`CVE-2024-5321` .
`1.26-2024.06.17`	`1.26.15`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.26-2024.05.14`	`1.26.15`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.26.15`.
`1.26-2024.04.09`	`1.26.12`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.26-2024.03.12`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.02.13`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.01.09`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2023.12.12`	`1.26.10`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.11.14`	`1.26.10`	`1.6.18`	`1.1.2`	Comprend des correctifs pour `CVE-2023-5528` .
`1.26-2023.10.19`	`1.26.9`	`1.6.18`	`1.1.2`	Mise à niveau `containerd` vers `1.6.18`. Mise à niveau `kubelet` vers `1.26.9`. Ajout de nouvelles variables d'environnement pour le script d'amorçage (`SERVICE_IPV4_CIDR` et `EXCLUDED_SNAT_CIDRS`).

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.06.20`	`1.26.4`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.26-2023.06.14`	`1.26.4`	`1.6.6`	`1.1.1`	Mise à niveau de Kubernetes vers. `1.26.4` Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.05.09`	`1.26.2`	`1.6.6`	`1.1.1`	Correction d'un bogue provoquant le problème de connectivité réseau #1126 sur les pods après le redémarrage du nœud. Ajout d'un nouveau paramètre de configuration du script d'amorçage (`ExcludedSnatCIDRs`).
`1.26-2023.04.26`	`1.26.2`	`1.6.6`	`1.1.1`	
`1.26-2023.04.11`	`1.26.2`	`1.6.6`	`1.1.1`	Ajout d'un mécanisme de récupération pour `kubelet` et `kube-proxy` en cas de panne de service.
`1.26-2023.03.24`	`1.26.2`	`1.6.6`	`1.1.1`	

AMI Windows Server 2019 Core optimisée pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI Windows Server 2019 Core optimisée pour Amazon EKS.

Example

Version 1.33 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.33-2025.07.16`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.06.13`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.05.17`	`1.33.1`	`1.7.27`	`1.2.1`	

Version 1.32 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.32-2025.07.16`	`1.32.5`	`1.7.27`	`1.2.1`	
`1.32-2025.06.13`	`1.32.5`	`1.7.27`	`1.2.1`	Mise à niveau `containerd` vers `1.7.27`.
`1.32-2025.05.17`	`1.32.5`	`1.7.20`	`1.2.1`	
`1.32-2025.04.14`	`1.32.1`	`1.7.20`	`1.1.3`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.4	1.7.20	1.1.3	

Version 1.31 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.30-2025-02-15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau containerd vers 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau containerd vers 1.7.20.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.29-2024.05.15`	`1.29.3`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`. Mise à niveau `kubelet` vers `1.29.3`.
`1.29-2024.04.09`	`1.29.0`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.29-2024.03.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Correction d'un bug à cause duquel l'image de pause était supprimée par erreur par le processus de collecte `kubelet` des déchets.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Version 1.28 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.20	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Mise à niveau containerd vers 1.6.28. Mise à niveau kubelet vers 1.28.8.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Mise à niveau containerd vers 1.6.25. CNI reconstruit et csi-proxy utilisation golang 1.22.1.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Version 1.27 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2024.08.13`	`1.27.15`	`1.7.14`	`1.1.3`	
`1.27-2024.07.10`	`1.27.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour `CVE-2024-5321` .
`1.27-2024.06.17`	`1.27.12`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.27-2024.05.14`	`1.27.12`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.27.12`.
`1.27-2024.04.09`	`1.27.9`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang` `1.22.1`.
`1.27-2024.03.13`	`1.27.9`	`1.6.18`	`1.1.2`	
`1.27-2024.02.13`	`1.27.9`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2023.06.20`	`1.27.1`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.27-2023.06.14`	`1.27.1`	`1.6.6`	`1.1.1`	Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.
`1.27-2023.06.06`	`1.27.1`	`1.6.6`	`1.1.1`	Correction du problème #2042 `containers-roadmap` qui empêchait les nœuds d'extraire des images Amazon ECR privées.
`11.27-2023.05.18`	`1.27.1`	`1.6.6`	`1.1.1`	

Version 1.26 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2025.05.17`	`1.26.15`	`1.7.20`	`1.2.1`	
`1.26-2025.04.14`	`1.26.15`	`1.7.20`	`1.1.3`	
`1.26-2025.03.14`	`1.26.15`	`1.7.20`	`1.1.3`	Mise à niveau `containerd` vers `1.7.20`.
`1.26-2025.02.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2025.01.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.12.11`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.11.12`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.10.09`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.09.10`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.08.13`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.07.10`	`1.26.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
				`CVE-2024-5321` .
`1.26-2024.06.17`	`1.26.15`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.26-2024.05.14`	`1.26.15`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.26.15`.
`1.26-2024.04.09`	`1.26.12`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.26-2024.03.13`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.02.13`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.01.09`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2023.12.12`	`1.26.10`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.11.14`	`1.26.10`	`1.6.18`	`1.1.2`	Comprend des correctifs pour `CVE-2023-5528` .
`1.26-2023.10.19`	`1.26.9`	`1.6.18`	`1.1.2`	Mise à niveau `containerd` vers `1.6.18`. Mise à niveau `kubelet` vers `1.26.9`. Ajout de nouvelles variables d'environnement pour le script d'amorçage (`SERVICE_IPV4_CIDR` et `EXCLUDED_SNAT_CIDRS`).

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.06.20`	`1.26.4`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.26-2023.06.14`	`1.26.4`	`1.6.6`	`1.1.1`	Mise à niveau de Kubernetes vers. `1.26.4` Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.05.09`	`1.26.2`	`1.6.6`	`1.1.1`	Correction d'un bogue provoquant le problème de connectivité réseau #1126 sur les pods après le redémarrage du nœud. Ajout d'un nouveau paramètre de configuration du script d'amorçage (`ExcludedSnatCIDRs`).
`1.26-2023.04.26`	`1.26.2`	`1.6.6`	`1.1.1`	
`1.26-2023.04.11`	`1.26.2`	`1.6.6`	`1.1.1`	Ajout d'un mécanisme de récupération pour `kubelet` et `kube-proxy` en cas de panne de service.
`1.26-2023.03.24`	`1.26.2`	`1.6.6`	`1.1.1`	

AMI Windows Server 2019 Full optimisée pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI complète Windows Server 2019 optimisée pour Amazon EKS.

Example

Version 1.33 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.33-2025.07.16`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.06.13`	`1.33.1`	`1.7.27`	`1.2.1`	
`1.33-2025.05.17`	`1.33.1`	`1.7.27`	`1.2.1`	

Version 1.32 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.32-2025.07.16`	`1.32.5`	`1.7.27`	`1.2.1`	
`1.32-2025.06.13`	`1.32.5`	`1.7.27`	`1.2.1`	Mise à niveau `containerd` vers `1.7.27`.
`1.32-2025.05.17`	`1.32.5`	`1.7.20`	`1.2.1`	
`1.32-2025.04.14`	`1.32.1`	`1.7.20`	`1.1.3`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Version 1.31 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau containerd vers 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau containerd vers 1.7.20.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.29-2024.05.15`	`1.29.3`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`. Mise à niveau `kubelet` vers `1.29.3`.
`1.29-2024.04.09`	`1.29.0`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.29-2024.03.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.13`	`1.29.0`	`1.6.25`	`1.1.2`	
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Correction d'un bug à cause duquel l'image de pause était supprimée par erreur par le processus de collecte `kubelet` des déchets.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Version 1.28 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.28-2024.12.11`	`1.28.15`	`1.7.14`	`1.1.3`	
`1.28-2024.11.12`	`1.28.13`	`1.7.14`	`1.1.3`	
`1.28-2024.10.08`	`1.28.13`	`1.7.14`	`1.1.3`	
`1.28-2024.09.10`	`1.28.11`	`1.7.14`	`1.1.3`	
`1.28-2024.08.13`	`1.28.11`	`1.7.14`	`1.1.3`	
`1.28-2024.07.10`	`1.28.11`	`1.7.11`	`1.1.2`	Comprend des correctifs pour `CVE-2024-5321` .
`1.28-2024.06.17`	`1.28.8`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.28-2024.05.14`	`1.28.8`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.28.8`.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Mise à niveau containerd vers 1.6.25. CNI reconstruit et csi-proxy utilisation golang 1.22.1.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Version 1.27 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.07.16	1.27.16	1.7.27	1.2.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2025.06.13	1.27.16	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.27-2025.05.17	1.27.16	1.7.20	1.2.1	
1.27-2025.04.14	1.27.16	1.7.20	1.1.3	
1.27-2025.03.14	1.27.16	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.27-2025.02.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.15	1.27.16	1.7.14	1.1.3	
1.27-2025.01.01	1.27.16	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.27-2024.12.11	1.27.16	1.7.14	1.1.3	
1.27-2024.11.12	1.27.16	1.7.14	1.1.3	
1.27-2024.10.08	1.27.16	1.7.14	1.1.3	
1.27-2024.09.10	1.27.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2024.08.13`	`1.27.15`	`1.7.14`	`1.1.3`	
`1.27-2024.07.10`	`1.27.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour `CVE-2024-5321` .
`1.27-2024.06.17`	`1.27.12`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.27-2024.05.14`	`1.27.12`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.27.12`.
`1.27-2024.04.09`	`1.27.9`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang` `1.22.1`.
`1.27-2024.03.13`	`1.27.9`	`1.6.18`	`1.1.2`	
`1.27-2024.02.13`	`1.27.9`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.27-2023.06.20`	`1.27.1`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.27-2023.06.14`	`1.27.1`	`1.6.6`	`1.1.1`	Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.
`1.27-2023.06.06`	`1.27.1`	`1.6.6`	`1.1.1`	Correction du problème #2042 `containers-roadmap` qui empêchait les nœuds d'extraire des images Amazon ECR privées.
`1.27-2023.05.17`	`1.27.1`	`1.6.6`	`1.1.1`	

Version 1.26 de Kubernetes

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2025.05.17`	`1.26.15`	`1.7.20`	`1.2.1`	
`1.26-2025.04.14`	`1.26.15`	`1.7.20`	`1.1.3`	
`1.26-2025.03.14`	`1.26.15`	`1.7.20`	`1.1.3`	Mise à niveau `containerd` vers `1.7.20`.
`1.26-2025.02.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2025.01.15`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.12.11`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.11.12`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.10.08`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.09.10`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.08.13`	`1.26.15`	`1.7.14`	`1.1.3`	
`1.26-2024.07.10`	`1.26.15`	`1.7.11`	`1.1.2`	Comprend des correctifs pour

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
				`CVE-2024-5321` .
`1.26-2024.06.17`	`1.26.15`	`1.7.11`	`1.1.2`	Mise à niveau `containerd` vers `1.7.11`.
`1.26-2024.05.14`	`1.26.15`	`1.6.28`	`1.1.2`	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.26.15`.
`1.26-2024.04.09`	`1.26.12`	`1.6.25`	`1.1.2`	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilisation `golang 1.22.1`.
`1.26-2024.03.13`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.02.13`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2024.01.09`	`1.26.12`	`1.6.18`	`1.1.2`	
`1.26-2023.12.12`	`1.26.10`	`1.6.18`	`1.1.2`	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.11.14`	`1.26.10`	`1.6.18`	`1.1.2`	Comprend des correctifs pour `CVE-2023-5528` .
`1.26-2023.10.19`	`1.26.9`	`1.6.18`	`1.1.2`	Mise à niveau `containerd` vers `1.6.18`. Mise à niveau `kubelet` vers `1.26.9`. Ajout de nouvelles variables d'environnement pour le script d'amorçage (`SERVICE_IPV4_CIDR` et `EXCLUDED_SNAT_CIDRS`).

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Mise à niveau du plug-in Amazon VPC CNI pour utiliser le binaire du connecteur Kubernetes, qui obtient l'adresse IP du pod depuis le serveur d'API Kubernetes. Demande d'extraction n° 100 fusionnée.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	Comprend des correctifs pour CVE-2023-3676 , CVE-2023-3893 et CVE-2023-3955 .
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
`1.26-2023.06.20`	`1.26.4`	`1.6.6`	`1.1.1`	Résolution d'un problème qui entraînait le remplissage incorrect de la liste de recherche des suffixes DNS.
`1.26-2023.06.14`	`1.26.4`	`1.6.6`	`1.1.1`	Mise à niveau de Kubernetes vers. `1.26.4` Ajout du support pour le mappage des ports hôte dans CNI. Demande d'extraction fusionnée #93.

Version d'AMI	version kubelet	version containerd	version du proxy csi	Notes de mise à jour
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	Correction d'un bogue provoquant le problème de connectivité réseau #1126 sur les pods après le redémarrage du nœud. Ajout d'un nouveau paramètre de configuration du script d'amorçage (ExcludedSnatCIDRs).
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	Ajout d'un mécanisme de récupération pour kubelet et kube-proxy en cas de panne de service.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Récupérez l'AMI Microsoft Windows recommandée IDs

Lorsque vous déployez des nœuds, vous pouvez spécifier un ID pour une Amazon Machine Image (AMI) prédéfinie optimisée pour Amazon EKS. Pour récupérer un ID d'AMI correspondant à la configuration souhaitée, interrogez l'API du magasin de paramètres de AWS Systems Manager. L'utilisation de cette API élimine le besoin de rechercher manuellement l'AMI optimisée pour Amazon EKS IDs. Pour de plus amples informations, veuillez consulter GetParameter. Le principal IAM que vous utilisez doit disposer de l'autorisation IAM `ssm:GetParameter` pour récupérer les métadonnées de l'AMI optimisée pour Amazon EKS.

Vous pouvez récupérer l'ID d'image de la dernière AMI Windows optimisée pour Amazon EKS recommandée à l'aide de la commande suivante, qui utilise le sous-paramètre`image_id`. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- *`release`*Remplacez-le par l'une des options suivantes.
  - À utiliser *`2022`* pour Windows Server 2022.
  - À utiliser *`2019`* pour Windows Server 2019.
- *`installation-option`*Remplacez-le par l'une des options suivantes. Pour plus d'informations, voir Qu'est-ce que l'option d'installation Server Core dans Windows Server ?
  - *`Core`*À utiliser pour une installation minimale avec une surface d'attaque plus petite.
  - *`Full`*À utiliser pour inclure l'expérience de bureau Windows.
- *`kubernetes-version`*Remplacez-le par un eks/latest/userguide/platform fichier -versions.html compatible [platform-version, type="documentation "].
- *`region-code`*Remplacez-le par une AWS région prise en charge par Amazon EKS pour laquelle vous souhaitez obtenir l'ID AMI.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-
English-installation-option-EKS_Optimized-kubernetes-version/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Voici un exemple de commande après le remplacement des espaces réservés.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-
English-Core-EKS_Optimized-k8s-n-2/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

L'exemple qui suit illustre un résultat.

```
ami-1234567890abcdef0
```

Créez une AMI Windows personnalisée avec Image Builder

Vous pouvez utiliser EC2 Image Builder pour créer des fenêtres personnalisées optimisées pour Amazon EKS AMIs avec l'une des options suivantes :

- Utilisation d'une AMI Windows optimisée pour Amazon EKS comme base
- Utilisation du composant de compilation géré par Amazon

Vous devez créer votre propre recette Image Builder avec les deux méthodes. Pour plus d'informations, veuillez consulter la rubrique Créer une nouvelle version d'une recette d'image dans le Guide de l'utilisateur Image Builder.

> **Important**
>
> Les composants gérés par Amazon suivants pour `eks` incluent des correctifs pour `CVE-2024-5321`.
>
> - `1.26.4` et ultérieures
> - `1.27.2` et ultérieures
> - `1.28.2` et ultérieures
> - `1.29.2` et ultérieures
> - `1.30.1` et ultérieures

Utilisation d'une AMI Windows optimisée pour Amazon EKS comme base

Cette option est la méthode recommandée pour créer votre Windows personnalisé AMIs. Les fenêtres optimisées pour Amazon EKS que AMIs nous proposons sont mises à jour plus fréquemment que le composant de compilation géré par Amazon.

1. Lancez une nouvelle recette Image Builder.
   a. Ouvrez la console EC2 Image Builder à l'adresse https://console.aws.amazon.com/imagebuilder.

   b. Dans le panneau de navigation de gauche, choisissez Recettes d'image.
   c. Choisissez Créer une recette d'image.
2. Dans la section Détails de la recette, saisissez un Nom et une Version.
3. Spécifiez l'ID de l'AMI Windows optimisée pour Amazon EKS dans la section Image de base.
   a. Choisissez Saisir un ID d'AMI personnalisé.
   b. Récupérez l'ID AMI correspondant à la version du système d'exploitation Windows dont vous avez besoin. Pour de plus amples informations, veuillez consulter the section called "Recevez les dernières nouvelles IDs".
   c. Saisissez l'ID d'AMI personnalisé. Si l'ID AMI n'est pas trouvé, assurez-vous que la AWS région correspondant à l'ID d'AMI correspond à la AWS région affichée dans le coin supérieur droit de votre console.
4. (Facultatif) Pour obtenir les dernières mises à jour de sécurité, ajoutez le composant `update-windows` dans la section Composants de génération.
   a. Dans la liste déroulante située à droite du champ de recherche Rechercher des composants par nom, sélectionnez Géré par Amazon.
   b. Dans le champ de recherche Rechercher des composants par nom, saisissez `update-windows`.
   c. Cochez la case correspondant au résultat de la **`update-windows`**recherche. Ce composant inclut les derniers correctifs Windows pour le système d'exploitation.
5. Complétez les entrées de recette d'image restantes avec les configurations requises. Pour plus d'informations, veuillez consulter la rubrique Créer une nouvelle version d'une recette d'image (console) dans le Guide de l'utilisateur Image Builder.
6. Choisissez Créer une recette.
7. Utilisez la nouvelle recette d'image dans un pipeline d'images nouveau ou existant. Une fois que votre pipeline d'images s'exécute correctement, votre AMI personnalisée sera répertoriée en tant qu'image de sortie et sera prête à être utilisée. Pour plus d'informations, voir Création d'un pipeline d'images à l'aide de l'assistant de console EC2 Image Builder.

Utilisation du composant de génération géré par Amazon

Lorsque l'utilisation d'une AMI Windows optimisée pour Amazon EKS comme base n'est pas viable, vous pouvez utiliser le composant de génération géré par Amazon à la place. Cette option peut être en retard par rapport aux versions les plus récentes de Kubernetes prises en charge.

1. Lancez une nouvelle recette Image Builder.
   a. Ouvrez la console EC2 Image Builder à l'adresse https://console.aws.amazon.com/imagebuilder.
   b. Dans le panneau de navigation de gauche, choisissez Recettes d'image.
   c. Choisissez Créer une recette d'image.
2. Dans la section Détails de la recette, saisissez un Nom et une Version.
3. Déterminez l'option que vous utiliserez pour créer votre AMI personnalisée dans la section Image de base :
   - Sélectionnez les images gérées : choisissez Windows pour votre Système d'exploitation (OS) de l'image. Choisissez ensuite l'une des options suivantes pour Origine de l'image.
     - Démarrage rapide (géré par Amazon) : dans le menu déroulant Nom de l'image, choisissez une version de Windows Server compatible avec Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Windows".
     - Mes images : pour Nom de l'image, sélectionnez l'ARN de votre propre image avec votre propre licence. Les composants Amazon EKS ne peuvent pas déjà être installés sur l'image que vous fournissez.
   - Saisir l'ID d'AMI personnalisée : pour l'ID D'AMI, saisissez l'ID de votre AMI avec votre propre licence. Les composants Amazon EKS ne peuvent pas déjà être installés sur l'image que vous fournissez.
4. Dans la section Composants de génération – Windows, procédez comme suit :
   a. Dans la liste déroulante située à droite du champ de recherche Rechercher des composants par nom, sélectionnez Géré par Amazon.
   b. Dans le champ de recherche Rechercher des composants par nom, saisissez `eks`.
   c. Cochez la case du résultat de la **`eks-optimized-ami-windows`**recherche, même si le résultat renvoyé n'est peut-être pas la version que vous souhaitez.
   d. Dans le champ de recherche Rechercher des composants par nom, saisissez `update-windows`.
   e. Cochez la case du résultat de recherche update-windows. Ce composant inclut les derniers correctifs Windows pour le système d'exploitation.
5. Dans la section Composants sélectionnés, procédez comme suit :
   a. Choisissez les options de version pour. **`eks-optimized-ami-windows`**
   b. Choisissez Spécifier la version du composant.

c. Dans le champ Version du composant, saisissez*version.x*, en *version* remplaçant par une version Kubernetes prise en charge. La saisie *x* d'une partie du numéro de version indique que vous devez utiliser la dernière version du composant qui correspond également à la partie de la version que vous avez explicitement définie. Faites attention à la sortie de la console, car elle vous indiquera si la version souhaitée est disponible en tant que composant géré. N'oubliez pas que les versions les plus récentes de Kubernetes peuvent ne pas être disponibles pour le composant de compilation. Pour plus d'informations sur les versions disponibles, consultez the section called "Récupération d'informations sur les versions des composants `eks-optimized-ami-windows`".

6. Complétez les entrées de recette d'image restantes avec les configurations requises. Pour plus d'informations, veuillez consulter la rubrique Créer une nouvelle version d'une recette d'image (console) dans le Guide de l'utilisateur Image Builder.
7. Choisissez Créer une recette.
8. Utilisez la nouvelle recette d'image dans un pipeline d'images nouveau ou existant. Une fois que votre pipeline d'images s'exécute correctement, votre AMI personnalisée sera répertoriée en tant qu'image de sortie et sera prête à être utilisée. Pour plus d'informations, voir Création d'un pipeline d'images à l'aide de l'assistant de console EC2 Image Builder.

Récupération d'informations sur les versions des composants **`eks-optimized-ami-windows`**

Vous pouvez récupérer des informations spécifiques concernant ce qui est installé avec chaque composant. Par exemple, vous pouvez vérifier quelle version de `kubelet` est installée. Les composants sont soumis à des tests fonctionnels sur les versions des systèmes d'exploitation Windows prises en charge par Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Calendrier des versions". Toute autre version du système d'exploitation Windows qui n'est pas répertoriée comme prise en charge ou dont le support a atteint la fin peut ne pas être compatible avec le composant.

1. Ouvrez la console EC2 Image Builder à l'adresse https://console.aws.amazon.com/imagebuilder.
2. Dans le panneau de navigation de gauche, sélectionnez Composants.
3. Dans la liste déroulante située à droite du champ de recherche Rechercher des composants par nom, modifiez M'appartenant en Démarrage rapide (géré par Amazon).
4. Dans la case Rechercher des composants par nom, saisissez `eks`.
5. (Facultatif) Si vous utilisez une version récente, triez la colonne Version par ordre décroissant en la choisissant deux fois.

6. Choisissez le **`eks-optimized-ami-windows`**lien avec la version souhaitée.

La Description de la page qui s'affiche contient les informations spécifiques.

Activez la réparation automatique des nœuds et étudiez les problèmes de santé des nœuds

L'état de santé du nœud fait référence à l'état opérationnel et à la capacité d'un nœud à exécuter efficacement les charges de travail. Un nœud sain maintient la connectivité attendue, dispose de ressources suffisantes et peut exécuter des Pods avec succès sans interruption. Pour plus d'informations sur l'obtention de détails sur vos nœuds, reportez-vous aux the section called "Afficher l'état du nœud" sections etthe section called "Obtenir les journaux des nœuds".

Pour aider à maintenir des nœuds sains, Amazon EKS propose l'agent de surveillance des nœuds et la réparation automatique des nœuds.

> **Important**
>
> L'agent de surveillance des nœuds et la réparation automatique des nœuds ne sont disponibles que sous Linux. Ces fonctionnalités ne sont pas disponibles sous Windows.

Agent de surveillance des nœuds

L'agent de surveillance des nœuds lit automatiquement les journaux des nœuds pour détecter certains problèmes de santé. Il analyse les journaux des nœuds pour détecter les défaillances et affiche diverses informations d'état sur les nœuds de travail. Un code dédié `NodeCondition` est appliqué sur les nœuds de travail pour chaque catégorie de problèmes détectés, tels que les problèmes de stockage et de réseau. Les descriptions des problèmes de santé détectés sont disponibles dans le tableau de bord d'observabilité. Pour de plus amples informations, veuillez consulter the section called "Problèmes de santé des nœuds".

L'agent de surveillance des nœuds est inclus en tant que fonctionnalité pour tous les clusters en mode automatique Amazon EKS. Pour les autres types de clusters, vous pouvez ajouter l'agent de surveillance en tant que module complémentaire Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Création d'un module complémentaire".

Réparation automatique de nœuds

La réparation automatique des nœuds est une fonctionnalité supplémentaire qui surveille en permanence l'état des nœuds, réagit automatiquement aux problèmes détectés et remplace les nœuds lorsque cela est possible. Cela permet une disponibilité globale du cluster avec une intervention manuelle minimale. Si un contrôle de santé échoue, le nœud est automatiquement bouclé afin qu'aucun nouveau pod ne soit programmé sur le nœud.

À elle seule, la réparation automatique des nœuds peut réagir à l'`Ready`état des objets du nœud `kubelet` et de tout objet supprimé manuellement. Lorsqu'elle est associée à l'agent de surveillance des nœuds, la réparation automatique des nœuds peut réagir à un plus grand nombre de conditions qui ne seraient pas détectées autrement. Ces conditions supplémentaires incluent `KernelReady``NetworkingReady`, et`StorageReady`.

Cette restauration automatique des nœuds résout automatiquement les problèmes intermittents liés aux nœuds, tels que les échecs de connexion au cluster, l'absence de réponse des kubelets et l'augmentation du nombre d'erreurs liées aux accélérateurs (appareils). La fiabilité améliorée permet de réduire les temps d'arrêt des applications et d'améliorer le fonctionnement des clusters. La réparation automatique du nœud ne peut pas gérer certains problèmes signalés`DiskPressure`, tels que`MemoryPressure`, et`PIDPressure`. Amazon EKS attend 10 minutes avant d'agir sur le `AcceleratedHardwareReadyNodeConditions`, et 30 minutes pour toutes les autres conditions.

Les groupes de nœuds gérés désactiveront également automatiquement les réparations de nœuds pour des raisons de sécurité dans deux scénarios. Toutes les opérations de réparation déjà en cours se poursuivront dans les deux situations.

- Si un changement de zone pour votre cluster a été déclenché par le biais de l'Application Recovery Controller (ARC), toutes les opérations de réparation ultérieures sont interrompues.
- Si votre groupe de nœuds compte plus de cinq nœuds et que plus de 20 % des nœuds de votre groupe de nœuds sont en mauvais état, les opérations de réparation sont interrompues.

Vous pouvez activer la réparation automatique des nœuds lors de la création ou de la modification d'un groupe de nœuds géré.

- Lorsque vous utilisez la console Amazon EKS, cochez la case Activer la réparation automatique du nœud pour le groupe de nœuds géré. Pour de plus amples informations, veuillez consulter the section called “Création”.

- Lorsque vous utilisez la AWS CLI, ajoutez la `eks update-nodegroup-config`commande `--node-repair-config enabled=true` to the `eks create nodegroup`or.
- Pour un exemple d'utilisation `eksctl ClusterConfig` d'un groupe de nœuds géré avec réparation automatique des nœuds, consultez 44-node-repair.yaml on. GitHub

Problèmes de santé des nœuds

Les tableaux suivants décrivent les problèmes de santé des nœuds qui peuvent être détectés par l'agent de surveillance des nœuds. Il existe deux types de problèmes :

- Condition : problème de terminal qui justifie une action corrective telle que le remplacement ou le redémarrage d'une instance. Lorsque la réparation automatique est activée, Amazon EKS effectue une action de réparation, sous forme de remplacement du nœud ou de redémarrage. Pour de plus amples informations, veuillez consulter the section called "Conditions du nœud".
- Événement : problème temporaire ou configuration de nœud sous-optimale. Aucune action de réparation auto n'aura lieu. Pour de plus amples informations, veuillez consulter the section called "Événements liés aux nœuds".

Problèmes de santé des nœuds du noyau

Nom	Sévérité	Description
ForkFailedOutOfPID	Condition	Un appel fork ou exec a échoué en raison d'un manque de processus IDs ou de mémoire du système, ce qui peut être dû à des processus zombies ou à un épuisement physique de la mémoire.
AppBlocked	Événement	La planification de la tâche a été bloquée pendant une longue période, généralem ent en raison d'un blocage en entrée ou en sortie.

Nom	Sévérité	Description
AppCrash	Événement	Une application sur le nœud est tombée en panne.
ApproachingKernelPidMax	Événement	Le nombre de processus approche le nombre maximum de processus disponibles PIDs selon le paramètre kernel.pid_max actuel, après quoi aucun autre processus ne pourra être lancé.
ApproachingMaxOpenFiles	Événement	Le nombre de fichiers ouverts est proche du nombre maximum de fichiers ouverts possibles compte tenu des paramètres actuels du noyau, après quoi l'ouverture de nouveaux fichiers échouera.
ConntrackExceededKernel	Événement	Le suivi des connexions a dépassé le maximum pour le noyau et de nouvelles connexions n'ont pas pu être établies, ce qui peut entraîner une perte de paquets.
ExcessiveZombieProcesses	Événement	Les processus qui ne peuvent pas être entièrement récupérés s'accumulent en grand nombre, ce qui indique des problèmes d'application et peut conduire à l'atteinte des limites des processus du système.

Nom	Sévérité	Description
KernelBug	Événement	Un bogue du noyau a été détecté et signalé par le noyau Linux lui-même, bien que cela puisse parfois être dû à une utilisation élevée du processeur ou de la mémoire sur des nœuds, ce qui a entraîné un retard dans le traitement des événements.
LargeEnvironment	Événement	Le nombre de variables d'environnement associées à ce processus est supérieur aux prévisions, ce qui peut être dû au fait que de nombreux services sont enableServiceLinks définis sur true, ce qui peut entraîner des problèmes de performances.
RapidCron	Événement	Une tâche cron s'exécute plus rapidement que toutes les cinq minutes sur ce nœud, ce qui peut avoir un impact sur les performances si la tâche consomme des ressources importantes.
SoftLockup	Événement	Le processeur s'est arrêté pendant un certain temps.

Problèmes de santé des nœuds réseau

Nom	Sévérité	Description
InterfaceNotRunning	Condition	Cette interface ne semble pas fonctionner ou il y a des problèmes de réseau.
InterfaceNotUp	Condition	Cette interface ne semble pas fonctionner ou il y a des problèmes de réseau.
IPAMDNotPrêt	Condition	L'IPAMD ne parvient pas à se connecter au serveur API.
IPAMDNotCourir	Condition	Le `aws-k8s-agent` processus n'a pas été détecté comme étant en cours d'exécution.
MissingLoopbackInterface	Condition	L'interface de boucle est absente de cette instance, ce qui entraîne une défaillance des services en fonction de la connectivité locale.
BandwidthInExceeded	Événement	Des paquets ont été mis en file d'attente ou supprimés car la bande passante agrégée entrante dépassait le maximum pour l'instance.
BandwidthOutExceeded	Événement	Des paquets ont été mis en file d'attente ou supprimés car la bande passante agrégée sortante dépassait le maximum pour l'instance.

Nom	Sévérité	Description
ConntrackExceeded	Événement	Le suivi des connexions a dépassé le maximum pour l'instance et aucune nouvelle connexion n'a pu être établie, ce qui peut entraîner une perte de paquets.
IPAMDNoIPs	Événement	Il n'y a plus d'adresses IP à l'IPAM-D.
IPAMDRepeatedlyRedémarrer	Événement	Plusieurs redémarrages du service IPAMD ont eu lieu.
KubeProxyNotReady	Événement	Kube-proxy n'a pas réussi à surveiller ou à répertorier les ressources.
LinkLocalExceeded	Événement	Les paquets ont été abandonnés car le PPS du trafic vers les services proxy locaux dépassait le maximum de l'interface réseau.
MissingDefaultRoutes	Événement	Les règles d'itinéraire par défaut sont manquantes.
IPRulesDisparus, disparus IPRoutes	Événement	Des règles de routage sont absentes de la table IPs de routage pour le pod suivant.
NetworkSysctl	Événement	Les paramètres réseau sysctl de ce nœud sont potentiel lement incorrects.

Nom	Sévérité	Description
PortConflict	Événement	Si un Pod utilise HostPort, il peut écrire des règles iptables qui remplacent les ports déjà liés de l'hôte, empêchant potentiellement l'accès du serveur API à. `kubelet`
PPSExceeded	Événement	Des paquets ont été mis en file d'attente ou supprimés car le PPS bidirectionnel a dépassé le maximum pour l'instance.
UnexpectedRejectRule	Événement	Une `DROP` règle inattendue `REJECT` a été détectée dans les iptables, bloquant potentiel lement le trafic attendu.

Problèmes de santé des nœuds neuronaux

Nom	Sévérité	Description
Neurone DMAError	Condition	Un moteur DMA a rencontré une erreur irrécupérable.
Erreur neuronale HBMUncorr ectable	Condition	Un HBM a rencontré une erreur irréparable et a produit des résultats incorrects.
Erreur neuronale NCUncorre ctable	Condition	Une erreur de mémoire non corrigible de Neuron Core a été détectée.
Erreur neuronale SRAMUncor rectable	Condition	Une mémoire SRAM intégrée a rencontré une erreur

Nom	Sévérité	Description
		de parité et a produit des résultats incorrects.

Problèmes de santé des nœuds NVIDIA

Si la réparation automatique est activée, les actions de réparation répertoriées démarrent 10 minutes après la détection du problème. Pour plus d'informations sur les erreurs XID, consultez la section Erreurs Xid dans la documentation de déploiement et de gestion des GPU NVIDIA. Pour plus d'informations sur les messages XID individuels, consultez la section Comprendre les messages Xid dans la documentation de déploiement et de gestion des GPU NVIDIA.

Nom	Sévérité	Description	Action de réparation
NvidiaDoubleBitError	Condition	Une erreur de double bit a été produite par le pilote du GPU.	Remplacez
NVLinkErreur Nvidia	Condition	NVLink des erreurs ont été signalées par le pilote du GPU.	Remplacez
XID13Erreur Nvidia	Condition	Il existe une exception liée au moteur graphique.	Redémarrer
XID31Erreur Nvidia	Condition	Des problèmes matériels sont suspectés.	Redémarrer
XID48Erreur Nvidia	Condition	Les erreurs ECC à double bit sont signalées par le pilote.	Redémarrer
XID63Erreur Nvidia	Condition	Il y a un retrait de page ou une modificat	Redémarrer

Nom	Sévérité	Description	Action de réparation
		ion du mappage des lignes.	
XID64Erreur Nvidia	Condition	La tentative de retrait d'une page ou de remappage d'un nœud échoue.	Redémarrer
XID74Erreur Nvidia	Condition	Un problème est survenu lors de la connexion entre le processeur graphique et un autre processeu r graphique ou NVSwitch supérieur NVLink. Cela peut indiquer une défaillan ce matérielle du lien lui-même ou un problème lié à l'appareil situé à l'extrémité distante du lien.	Remplacez
XID79Erreur Nvidia	Condition	Le pilote du GPU a tenté d'accéder au GPU via sa connexion PCI Express et a constaté que le GPU n'était pas accessible.	Remplacez
XID94Erreur Nvidia	Condition	Il existe des erreurs de mémoire ECC.	Redémarrer
XID95Erreur Nvidia	Condition	Il existe des erreurs de mémoire ECC.	Redémarrer

Nom	Sévérité	Description	Action de réparation
XID119Erreur Nvidia	Condition	Le GSP a expiré pour répondre aux demandes RPC provenant d'autres bits du pilote.	Remplacez
Erreur Nvidia XID12 0	Condition	Le GSP a répondu à temps, mais avec une erreur.	Remplacez
XID121Erreur Nvidia	Condition	C2C est une interconnexion de puces. Il permet de partager de la mémoire entre CPUs des accélérateurs, etc.	Remplacez
Erreur Nvidia XID14 0	Condition	Le pilote du processeur graphique a peut-être observé des erreurs irréparables dans la mémoire du processeur graphique, de manière à interrompre la capacité du pilote du processeur graphique à marquer les pages pour le délignage dynamique des pages ou le remappage des lignes.	Remplacez

Nom	Sévérité	Description	Action de réparation
NvidiaPageRetirement	Événement	Le pilote du GPU a marqué une page de mémoire pour suppression. Cela peut se produire s'il y a une seule erreur sur deux bits ou si deux erreurs sur un seul bit sont rencontrées à la même adresse.	Aucun
Avertissement NvidiaXid [Code]	Événement	Toute occurrence XIDs autre que celles définies dans cette liste entraîne cet événement.	Aucun
DCGMError	Condition	La connexion au processus hôte du Data Center GPU Manager (DCGM) a été perdue ou n'a pas pu être établie.	Aucun
DCGMDiagnosticErreur	Condition	Un problème s'est produit lors de l'exécution des diagnostics actifs du DCGM.	Aucun
DCGMDiagnosticDéfaillance	Condition	Un scénario de test de la suite de tests DCGM Active Diagnostics a échoué.	Aucun

Problèmes de santé du nœud d'exécution

Nom	Sévérité	Description
PodStuckTerminating	Condition	Un pod est ou a été bloqué pendant trop longtemps, ce qui peut être dû à des erreurs CRI empêchant la progression de l'état du pod.
%sRepeatedRestart	Événement	Redémarre n'importe quel service systemd sur le nœud (formaté à l'aide du nom de l'unité indiqué par le titre).
ContainerRuntimeFailed	Événement	Le moteur d'exécution du conteneur n'a pas réussi à créer de conteneur, probablem ent en raison de problèmes signalés s'ils se produisaient à plusieurs reprises.
KubeletFailed	Événement	Le kubelet est entré dans un état d'échec.
LivenessProbeFailures	Événement	Une défaillance de la sonde de réactivité a été détectée, ce qui peut indiquer des problèmes de code d'applica tion ou des valeurs de délai d'expiration insuffisantes en cas de survenue répétée.
ReadinessProbeFailures	Événement	Une défaillance de la sonde de disponibilité a été détectée, ce qui peut indiquer des problèmes liés au code de l'application ou des valeurs de

Nom	Sévérité	Description
		délai d'expiration insuffisantes en cas de répétition.
ServiceFailedToStart	Événement	Une unité systemd n'a pas pu démarrer.

Problèmes de santé des nœuds de stockage

Nom	Sévérité	Description
XFSSmallAverageClusterSize	Condition	La taille moyenne du cluster XFS est petite, ce qui indique une fragmentation excessive de l'espace libre susceptib le d'empêcher la création de fichiers malgré les inodes ou l'espace libre disponibles.
EtcHostsMountFailed	Événement	Le montage du kubelet généré `/etc/hosts` a échoué en raison du remontage des données utilisateur pendant le fonctionnement du kubelet-container`/var/lib/kubelet/pods` .
IODelays	Événement	Retard d'entrée ou de sortie détecté dans un processus , indiquant potentiellement un approvisionnement en entrées-sorties insuffisant s'il est excessif.
KubeletDiskUsageSlow	Événement	Kubelet signale une utilisati on lente du disque lors de la

Nom	Sévérité	Description
		tentative d'accès au système de fichiers, ce qui peut indiquer une insuffisance des entrées-sorties du disque ou des problèmes de système de fichiers.

Afficher l'état de santé de vos nœuds

Cette rubrique décrit les outils et méthodes disponibles pour surveiller l'état de santé des nœuds dans les clusters Amazon EKS. Les informations couvrent l'état des nœuds, les événements et les cas de détection qui vous aident à identifier et à diagnostiquer les problèmes au niveau des nœuds. Utilisez les commandes et les modèles décrits ici pour inspecter les ressources de santé des nœuds, interpréter les conditions d'état et analyser les événements des nœuds à des fins de résolution des problèmes opérationnels.

Vous pouvez obtenir des informations sur l'état des nœuds à l'aide des commandes Kubernetes pour tous les nœuds. Et si vous utilisez l'agent de surveillance des nœuds via le mode automatique Amazon EKS ou le module complémentaire géré Amazon EKS, vous obtiendrez une plus grande variété de signaux de nœuds pour vous aider à résoudre les problèmes. Les descriptions des problèmes de santé détectés par l'agent de surveillance des nœuds sont également disponibles dans le tableau de bord d'observabilité. Pour de plus amples informations, veuillez consulter the section called “Santé des nœuds”.

Conditions du nœud

L'état des nœuds représente les problèmes du terminal nécessitant des mesures correctives telles que le remplacement ou le redémarrage d'une instance.

Pour obtenir les conditions de tous les nœuds, procédez comme suit :

```
kubectl get nodes -o 'custom-
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

Pour obtenir les conditions détaillées d'un nœud spécifique

```
kubectl describe node node-name
```

Exemple de sortie de condition d'un nœud sain :

```
  - lastHeartbeatTime: "2024-11-21T19:07:40Z"
    lastTransitionTime: "2024-11-08T03:57:40Z"
    message: Monitoring for the Networking system is active
    reason: NetworkingIsReady
    status: "True"
    type: NetworkingReady
```

Exemple d'état d'un nœud défectueux présentant un problème de réseau :

```
  - lastHeartbeatTime: "2024-11-21T19:12:29Z"
    lastTransitionTime: "2024-11-08T17:04:17Z"
    message: IPAM-D has failed to connect to API Server which could be an issue with
      IPTable rules or any other network configuration.
    reason: IPAMDNotReady
    status: "False"
    type: NetworkingReady
```

Événements liés aux nœuds

Les événements liés aux nœuds indiquent des problèmes temporaires ou des configurations sous-optimales.

Pour obtenir tous les événements signalés par l'agent de surveillance des nœuds

Lorsque l'agent de surveillance des nœuds est disponible, vous pouvez exécuter la commande suivante.

```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

Exemple de sortie :

```
LAST SEEN   TYPE      REASON       OBJECT
 MESSAGE
4s          Warning   SoftLockup   node/ip-192-168-71-251.us-west-2.compute.internal
 CPU stuck for 23s
```

Pour obtenir des événements pour tous les nœuds

```
kubectl get events --field-selector involvedObject.kind=Node
```

Pour obtenir des événements pour un nœud spécifique

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

Pour suivre les événements en temps réel

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

Exemple de sortie d'événement :

```
LAST SEEN   TYPE     REASON           OBJECT         MESSAGE
2m          Warning  MemoryPressure   Node/node-1    Node experiencing memory pressure
5m          Normal   NodeReady        Node/node-1    Node became ready
```

Commandes de dépannage courantes

```
# Get comprehensive node status
kubectl get node node-name -o yaml

# Watch node status changes
kubectl get nodes -w

# Get node metrics
kubectl top node
```

Récupérez les journaux d'un nœud géré à l'aide de kubectl et S3

Découvrez comment récupérer les journaux de nœuds pour un nœud géré par Amazon EKS doté de l'agent de surveillance des nœuds.

Prérequis

Vérifiez que vous disposez des éléments suivants :

- Un cluster Amazon EKS existant avec l'agent de surveillance des nœuds. Pour de plus amples informations, veuillez consulter the section called “Santé des nœuds”.

- L'outil de `kubectl` ligne de commande installé et configuré pour communiquer avec votre cluster.
- La AWS CLI s'est installée et s'est connectée avec des autorisations suffisantes pour créer des compartiments et des objets S3.
- Une version récente de Python 3 installée
- Le AWS SDK pour Python 3, Boto 3, est installé.

Étape 1 : créer une destination de compartiment S3 (facultatif)

Si vous ne possédez pas encore de compartiment S3 pour stocker les journaux, créez-en un. Utilisez la commande AWS CLI suivante. Le bucket utilise par défaut la liste de contrôle `private` d'accès. *`bucket-name`*Remplacez-le par le nom de compartiment unique que vous avez choisi.

```
aws s3api create-bucket --bucket bucket-name
```

Étape 2 : Création d'une URL S3 pré-signée pour HTTP Put

Amazon EKS renvoie les journaux des nœuds en effectuant une opération HTTP PUT sur une URL que vous spécifiez. Dans ce didacticiel, nous allons générer une URL HTTP PUT S3 pré-signée.

Les journaux seront renvoyés sous forme d'archive gzip, avec l'`.tar.gz`extension.

> **Note**
>
> Vous devez utiliser l' AWS API ou un SDK pour créer l'URL de téléchargement S3 pré-signée afin qu'EKS télécharge le fichier journal. Vous ne pouvez pas créer d'URL de téléchargement S3 pré-signée à l'aide de la AWS CLI.

1. Déterminez l'endroit du compartiment dans lequel vous souhaitez stocker les journaux. Par exemple, vous pouvez utiliser `2024-11-12/logs1.tar.gz` comme clé.
2. Enregistrez le code Python suivant dans le fichier`presign-upload.py`. Remplacez `<bucket-name>` et`<key>`. La clé doit se terminer par`.tar.gz`.

```
import boto3; print(boto3.client('s3').generate_presigned_url(
   ClientMethod='put_object',
   Params={'Bucket': '<bucket-name>', 'Key': '<key>'},
```

```
    ExpiresIn=1000
))
```

3. Exécutez le script avec

```
python presign-upload.py
```

4. Notez la sortie de l'URL. Utilisez cette valeur à l'étape suivante en tant que`http-put-destination`.

Pour plus d'informations, consultez Générer une URL présignée pour télécharger un fichier dans la documentation du SDK AWS Boto3 pour Python.

Étape 3 : Création d'une NodeDiagnostic ressource

Identifiez le nom du nœud à partir duquel vous souhaitez collecter les journaux.

Créez un `NodeDiagnostic` manifeste qui utilise le nom du nœud comme nom de la ressource et fournissant une destination URL HTTP PUT.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
    name: node-name
spec:
    logCapture:
        destination: http-put-destination
```

Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f nodediagnostic.yaml
```

Vous pouvez vérifier l'état de la collection en décrivant la `NodeDiagnostic` ressource :

- Le statut `Success` ou `SuccessWithErrors` indique que la tâche est terminée et que les journaux ont été téléchargés vers la destination indiquée (`SuccessWithErrors`indique que certains journaux sont peut-être manquants)
- Si le statut est Échec, vérifiez que l'URL de téléchargement est bien formée et qu'elle n'a pas expiré.

```
kubectl describe nodediagnostics.eks.amazonaws.com/node-name
```

Étape 4 : télécharger les journaux depuis S3

Patientez environ une minute avant de tenter de télécharger les journaux. Utilisez ensuite la CLI S3 pour télécharger les journaux.

```
# Once NodeDiagnostic shows Success status, download the logs
aws s3 cp s3://bucket-name/key ./node-logs.tar.gz
```

Étape 5 : Nettoyer les NodeDiagnostic ressources

- `NodeDiagnostic`les ressources ne sont pas automatiquement supprimées. Vous devez les nettoyer vous-même une fois que vous aurez obtenu vos artefacts de journal.

```
# Delete the NodeDiagnostic resource
kubectl delete nodediagnostics.eks.amazonaws.com/node-name
```

Présentation des nœuds hybrides Amazon EKS

Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et périphérique en tant que nœuds dans des clusters Amazon EKS. AWS gère le plan de contrôle Kubernetes AWS hébergé du cluster Amazon EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements locaux ou périphériques. Cela permet d'unifier la gestion de Kubernetes au sein de vos environnements et de déléguer la gestion du plan de contrôle Kubernetes à vos applications locales et périphériques. AWS

Amazon EKS Hybrid Nodes fonctionne avec n'importe quel matériel sur site ou machine virtuelle, apportant l'efficacité, l'évolutivité et la disponibilité d'Amazon EKS partout où vos applications doivent s'exécuter. Vous pouvez utiliser un large éventail de fonctionnalités Amazon EKS avec les nœuds hybrides Amazon EKS, notamment les modules complémentaires Amazon EKS, Amazon EKS Pod Identity, les entrées d'accès au cluster, les informations sur le cluster et le support étendu des versions de Kubernetes. Amazon EKS Hybrid Nodes s'intègre nativement à AWS des services tels que AWS Systems Manager, AWS IAM Roles Anywhere, Amazon Managed Service for Prometheus

et CloudWatch Amazon pour une surveillance, une journalisation et une gestion des identités centralisées.

Avec les nœuds hybrides Amazon EKS, il n'y a aucun engagement initial ni aucun frais minimum, et les ressources vCPU de vos nœuds hybrides vous sont facturées à l'heure lorsqu'ils sont connectés à vos clusters Amazon EKS. Pour plus d'informations sur les tarifs, consultez la section Tarification d'Amazon EKS.

Fonctionnalités

EKS Hybrid Nodes possède les fonctionnalités de haut niveau suivantes :

- Plan de contrôle Kubernetes géré : AWS gère le plan de contrôle Kubernetes AWS hébergé du cluster EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements locaux ou périphériques. Cela permet d'unifier la gestion de Kubernetes au sein de vos environnements et de déléguer la gestion du plan de contrôle Kubernetes à vos applications locales et périphériques. AWS En déplaçant le plan de contrôle Kubernetes vers AWS, vous pouvez conserver la capacité locale de vos applications et avoir la certitude que le plan de contrôle Kubernetes s'adapte à vos charges de travail.
- Expérience EKS cohérente : la plupart des fonctionnalités EKS sont prises en charge par les nœuds hybrides EKS pour une expérience EKS cohérente dans vos environnements sur site et cloud, notamment les modules complémentaires EKS, l'identité du pod EKS, les entrées d'accès au cluster, les informations sur le cluster, le support étendu des versions de Kubernetes, etc. Consultez the section called “Configuration des modules complémentaires” pour plus d'informations sur les modules complémentaires EKS pris en charge par les nœuds hybrides EKS.
- Observabilité et gestion des identités centralisées : EKS Hybrid Nodes s'intègre nativement à AWS des services tels que AWS Systems Manager, AWS IAM Roles Anywhere, Amazon Managed Service for Prometheus et Amazon CloudWatch pour une surveillance, une journalisation et une gestion des identités centralisées.
- Burst-to-cloud ou ajouter de la capacité sur site : un seul cluster EKS peut être utilisé pour exécuter des nœuds hybrides et des nœuds dans des AWS Régions, des Zones AWS Locales ou des AWS Outposts ou pour ajouter de la capacité sur site burst-to-cloud à vos clusters EKS. Voir Considérations relatives aux clusters en mode mixte pour plus d'informations.
- Infrastructure flexible : EKS Hybrid Nodes suit une approche « apportez votre propre infrastructure » et est indépendante de l'infrastructure que vous utilisez pour les nœuds hybrides. Vous pouvez exécuter des nœuds hybrides sur des machines physiques ou virtuelles, ainsi que sur

des architectures x86 et ARM, ce qui permet de migrer les charges de travail sur site exécutées sur des nœuds hybrides entre différents types d'infrastructure.

- Mise en réseau flexible : avec les nœuds hybrides EKS, la communication entre le plan de contrôle EKS et les nœuds hybrides est acheminée via le VPC et les sous-réseaux que vous transmettez lors de la création du cluster, en s'appuyant sur le mécanisme existant dans EKS pour la mise en réseau du plan de contrôle à nœud. Cela s'adapte à votre méthode préférée pour connecter vos réseaux locaux à un AWS VPC en entrée. Plusieurs options documentées sont disponibles, notamment AWS Site-to-Site VPN, AWS Direct Connect ou votre propre solution VPN, et vous pouvez choisir la méthode la mieux adaptée à votre cas d'utilisation.

Limites

- Jusqu'à 15 CIDRs pour les réseaux de nœuds distants et 15 CIDRs pour les réseaux de modules distants par cluster sont pris en charge.

Considérations

- Les nœuds hybrides EKS peuvent être utilisés avec des clusters EKS nouveaux ou existants.
- EKS Hybrid Nodes est disponible dans toutes les AWS régions, à l'exception des régions AWS GovCloud (États-Unis) et de la AWS Chine.
- Les nœuds hybrides EKS doivent disposer d'une connexion fiable entre votre environnement sur site et AWS. Les nœuds hybrides EKS ne sont pas adaptés aux environnements déconnectés, perturbés, intermittents ou limités (DDIL). Si vous utilisez un environnement DDIL, pensez à Amazon EKS Anywhere. Consultez les meilleures pratiques pour les nœuds hybrides EKS pour obtenir des informations sur le comportement des nœuds hybrides lors de scénarios de déconnexion du réseau.
- L'exécution de nœuds hybrides EKS sur une infrastructure cloud, notamment AWS dans des régions, AWS des zones locales, des AWS Outposts ou dans d'autres clouds, n'est pas prise en charge. Les frais relatifs aux nœuds hybrides vous seront facturés si vous exécutez des nœuds hybrides sur des EC2 instances Amazon.
- La facturation des nœuds hybrides commence lorsque les nœuds rejoignent le cluster EKS et s'arrête lorsque les nœuds sont retirés du cluster. Assurez-vous de supprimer vos nœuds hybrides de votre cluster EKS si vous ne les utilisez pas.

Ressources supplémentaires

- Atelier EKS Hybrid Step-by-step Nodes : instructions pour déployer les nœuds hybrides EKS dans un environnement de démonstration.
- AWS re:Invent : EKS Hybrid Nodes : session AWS re:Invent présentant le lancement d'EKS Hybrid Nodes avec un client expliquant comment il utilise les nœuds hybrides EKS dans son environnement.
- AWS Re:post : Mise en réseau de clusters pour les nœuds hybrides EKS : article expliquant les différentes méthodes de configuration de la mise en réseau pour les nœuds hybrides EKS.
- AWS blog : Exécutez l'inférence GenAI dans des environnements avec des nœuds hybrides EKS : article de blog expliquant comment exécuter l'inférence GenAI dans des environnements avec des nœuds hybrides EKS.

Configuration préalable pour les nœuds hybrides

Pour utiliser les nœuds hybrides Amazon EKS, vous devez disposer d'une connectivité privée depuis votre environnement sur site vers/depuis AWS, des serveurs bare metal ou des machines virtuelles dotés d'un système d'exploitation compatible, et des activations hybrides AWS IAM Roles Anywhere ou AWS Systems Manager (SSM) configurées. Vous êtes responsable de la gestion de ces prérequis tout au long du cycle de vie des nœuds hybrides.

- Connectivité réseau hybride depuis votre environnement sur site vers/depuis AWS
- Infrastructure sous forme de machines physiques ou virtuelles
- Système d'exploitation compatible avec les nœuds hybrides
- Fournisseur d'informations d'identification IAM sur site configuré

Connectivité réseau hybride

La communication entre le plan de contrôle Amazon EKS et les nœuds hybrides est acheminée via le VPC et les sous-réseaux que vous transmettez lors de la création du cluster, en s'appuyant sur le mécanisme existant dans Amazon EKS pour la mise en réseau du plan de contrôle à nœud. Plusieurs options documentées vous permettent de connecter votre environnement sur site à votre VPC, AWS Site-to-Site notamment le VPN, AWS Direct Connect ou votre propre connexion VPN. Consultez les guides d'utilisation du AWS Site-to-Site VPN et de AWS Direct Connect pour plus d'informations sur l'utilisation de ces solutions pour votre connexion réseau hybride.

Pour une expérience optimale, nous vous recommandons de disposer d'une connectivité réseau fiable d'au moins 100 Mbits/s et d'une latence aller-retour maximale de 200 ms pour la connexion des nœuds hybrides à la AWS région. Il s'agit d'une directive générale qui s'adapte à la plupart des cas d'utilisation, mais qui ne constitue pas une exigence stricte. Les exigences en matière de bande passante et de latence peuvent varier en fonction du nombre de nœuds hybrides et des caractéristiques de votre charge de travail, telles que la taille de l'image de l'application, l'élasticité de l'application, les configurations de surveillance et de journalisation, ainsi que les dépendances des applications en matière d'accès aux données stockées dans d'autres AWS services. Nous vous recommandons de tester avec vos propres applications et environnements avant le déploiement en production afin de vérifier que votre configuration réseau répond aux exigences de vos charges de travail.

Configuration réseau sur site

Vous devez activer l'accès réseau entrant depuis le plan de contrôle Amazon EKS vers votre environnement sur site pour permettre au plan de contrôle Amazon EKS de communiquer avec les `kubelet` nœuds hybrides et éventuellement avec les webhooks exécutés sur vos nœuds hybrides. En outre, vous devez activer l'accès au réseau sortant pour vos nœuds hybrides et les composants qui s'exécutent sur ceux-ci afin de communiquer avec le plan de contrôle Amazon EKS. Vous pouvez configurer cette communication pour qu'elle reste totalement privée avec votre AWS Direct Connect, votre AWS Site-to-Site VPN ou votre propre connexion VPN.

Les plages de routage interdomaines sans classe (CIDR) que vous utilisez pour vos réseaux de nœuds et de pods locaux doivent utiliser IPv4 des plages d'adresses RFC-1918. Votre routeur local doit être configuré avec des itinéraires vers vos nœuds locaux et, éventuellement, vos pods. Consultez the section called "Configuration réseau sur site" pour plus d'informations sur les exigences du réseau local, y compris la liste complète des ports et protocoles requis qui doivent être activés dans votre pare-feu et votre environnement sur site.

Configuration du cluster EKS

Pour minimiser la latence, nous vous recommandons de créer votre cluster Amazon EKS dans la AWS région la plus proche de votre environnement local ou périphérique. Vous transmettez votre nœud et votre pod locaux CIDRs lors de la création du cluster Amazon EKS via deux champs d'API : `RemoteNodeNetwork` et`RemotePodNetwork`. Vous devrez peut-être discuter avec votre équipe réseau sur site pour identifier votre nœud et votre pod sur site. CIDRs Le nœud CIDR est alloué depuis votre réseau sur site et le nœud CIDR est alloué depuis l'interface réseau de conteneurs (CNI) que vous utilisez si vous utilisez un réseau superposé pour votre CNI. Cilium et Calico utilisent des réseaux superposés par défaut.

Le nœud et le pod sur site CIDRs que vous configurez via les `RemotePodNetwork` champs `RemoteNodeNetwork` et sont utilisés pour configurer le plan de contrôle Amazon EKS afin d'acheminer le trafic via votre VPC vers `kubelet` les pods exécutés sur vos nœuds hybrides. Votre nœud et votre pod sur site CIDRs ne peuvent pas se chevaucher, ni avec le CIDR VPC que vous transmettez lors de la création du cluster, ni avec la configuration du IPv4 service pour votre cluster Amazon EKS.

Nous vous recommandons d'utiliser un accès au point de terminaison public ou privé pour le point de terminaison du serveur d'API Amazon EKS Kubernetes. Si vous choisissez « Public et privé », le point de terminaison du serveur d'API Amazon EKS Kubernetes indiquera toujours au public IPs les nœuds hybrides exécutés en dehors de votre VPC, ce qui peut empêcher vos nœuds hybrides

de rejoindre le cluster. Lorsque vous utilisez l'accès au point de terminaison public, le point de terminaison du serveur d'API Kubernetes devient public IPs et les communications entre les nœuds hybrides et le plan de contrôle Amazon EKS sont acheminées via Internet. Lorsque vous choisissez un accès de point de terminaison privé, le point de terminaison du serveur d'API Kubernetes devient privé IPs et les communications entre les nœuds hybrides et le plan de contrôle Amazon EKS seront acheminées via votre lien de connectivité privé, dans la plupart des cas Direct AWS Connect ou VPN. AWS Site-to-Site

Configuration VPC

Vous devez configurer le VPC que vous transmettez lors de la création du cluster Amazon EKS avec des itinéraires dans sa table de routage pour votre nœud local et éventuellement des réseaux de pods avec votre passerelle privée virtuelle (VGW) ou votre passerelle de transit (TGW) comme cible. Un exemple est illustré ci-dessous. Remplacez `REMOTE_NODE_CIDR` et `REMOTE_POD_CIDR` par les valeurs de votre réseau local.

Destination	Cible	Description
10,226,0,0/16	local	Le trafic local vers les routes du VPC au sein du VPC
REMOTE_NODE_CIDR	tgw-abcdef123456	Nœud CIDR sur site, route le trafic vers le TGW
REMODE_POD_CIDR	tgw-abcdef123456	Module CIDR sur site, acheminez le trafic vers le TGW

Configuration du groupe de sécurité

Lorsque vous créez un cluster, Amazon EKS crée un groupe de sécurité nommé`eks-cluster-sg-<cluster-name>-<uniqueID>`. Vous ne pouvez pas modifier les règles entrantes de ce groupe de sécurité du cluster, mais vous pouvez restreindre les règles sortantes. Vous devez ajouter un groupe de sécurité supplémentaire à votre cluster pour permettre au kubelet et éventuellement aux webhooks exécutés sur vos nœuds hybrides de contacter le plan de contrôle Amazon EKS. Les règles de trafic entrant requises pour ce groupe de sécurité supplémentaire sont indiquées ci-dessous. Remplacez `REMOTE_NODE_CIDR` et `REMOTE_POD_CIDR` par les valeurs de votre réseau local.

Nom	ID de règle du groupe de sécurité	Version IP	Type	Protocole	Plage de ports	Source
Nœud entrant sur site	sgr-abcdef123456	IPv4	HTTPS	TCP	443	REMOTE_NODE_CIDR
Envoi du pod sur site	sgr-abcdef654321	IPv4	HTTPS	TCP	443	REMOTE_POD_CIDR

Infrastructure

Vous devez disposer de serveurs bare metal ou de machines virtuelles pouvant être utilisés comme nœuds hybrides. Les nœuds hybrides sont indépendants de l'infrastructure sous-jacente et prennent en charge les architectures x86 et ARM. Amazon EKS Hybrid Nodes suit une approche « apportez votre propre infrastructure », dans le cadre de laquelle vous êtes responsable du provisionnement et de la gestion des serveurs bare metal ou des machines virtuelles que vous utilisez pour les nœuds hybrides. Bien qu'il n'y ait pas d'exigence minimale stricte en matière de ressources, nous vous recommandons d'utiliser des hôtes dotés d'au moins 1 vCPU et 1 Go de RAM pour les nœuds hybrides.

Système d'exploitation

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu et RHEL sont validés en permanence pour être utilisés comme système d'exploitation de nœuds pour les nœuds hybrides. Bottlerocket est uniquement pris en charge dans les environnements AWS vSphere VMware . AL2Le 023 n'est pas couvert par les plans de AWS Support lorsqu'il est exécuté en dehors d'Amazon EC2. AL2Le 023 ne peut être utilisé que dans des environnements virtualisés sur site. Consultez le guide de l'utilisateur Amazon Linux 2023 pour plus d'informations. AWS prend en charge l'intégration des nœuds hybrides avec les systèmes d'exploitation Ubuntu et RHEL, mais ne fournit pas de support pour le système d'exploitation lui-même.

Vous êtes responsable du provisionnement et de la gestion du système d'exploitation. Lorsque vous testez des nœuds hybrides pour la première fois, il est plus simple d'exécuter la CLI Amazon EKS Hybrid Nodes (`nodeadm`) sur un hôte déjà provisionné. Pour les déploiements de production,

nous vous recommandons d'inclure `nodeadm` dans votre système d'exploitation doré des images configurées pour s'exécuter en tant que service systemd afin de joindre automatiquement les hôtes aux clusters Amazon EKS au démarrage de l'hôte.

Fournisseur d'informations d'identification IAM sur site

Les nœuds hybrides Amazon EKS utilisent des informations d'identification IAM temporaires fournies par des activations hybrides AWS SSM ou par AWS IAM Roles Anywhere pour s'authentifier auprès du cluster Amazon EKS. Vous devez utiliser des activations hybrides AWS SSM ou des rôles AWS IAM Anywhere avec la CLI Amazon EKS Hybrid Nodes (). `nodeadm` Nous vous recommandons d'utiliser les activations hybrides AWS SSM si vous ne possédez pas d'infrastructure à clé publique (PKI) existante dotée d'une autorité de certification (CA) et de certificats pour vos environnements sur site. Si vous disposez d'une infrastructure PKI et de certificats sur site, utilisez AWS IAM Roles Anywhere.

Comme the section called "Rôle IAM de nœud" pour les nœuds exécutés sur Amazon EC2, vous allez créer un rôle IAM pour les nœuds hybrides avec les autorisations requises pour joindre des nœuds hybrides aux clusters Amazon EKS. Si vous utilisez AWS IAM Roles Anywhere, configurez une politique de confiance qui permet à AWS IAM Roles Anywhere d'assumer le rôle IAM de nœuds hybrides et configurez votre profil AWS IAM Roles Anywhere avec le rôle IAM de nœuds hybrides comme rôle assumé. Si vous utilisez AWS SSM, configurez une politique de confiance qui permet à AWS SSM d'assumer le rôle IAM des nœuds hybrides et de créer l'activation hybride avec le rôle IAM des nœuds hybrides. Découvrez the section called "Préparer les informations d'identification" comment créer le rôle IAM des nœuds hybrides avec les autorisations requises.

Préparer le réseau pour les nœuds hybrides

Cette rubrique fournit une vue d'ensemble de la configuration réseau que vous devez avoir configurée avant de créer votre cluster Amazon EKS et de connecter des nœuds hybrides. Ce guide part du principe que vous avez satisfait aux exigences requises pour la connectivité réseau hybride à l'aide d'AWS Site-to-Site un VPN, de AWS Direct Connect ou de votre propre solution VPN.

Configuration réseau sur site

Configuration réseau minimale requise

Pour une expérience optimale, nous vous recommandons de disposer d'une connectivité réseau fiable d'au moins 100 Mbits/s et d'une latence aller-retour maximale de 200 ms pour la connexion des nœuds hybrides à la AWS région. Il s'agit d'une directive générale qui s'adapte à la plupart des cas d'utilisation, mais qui ne constitue pas une exigence stricte. Les exigences en matière de bande passante et de latence peuvent varier en fonction du nombre de nœuds hybrides et des caractéristiques de votre charge de travail, telles que la taille de l'image de l'application, l'élasticité de l'application, les configurations de surveillance et de journalisation, et les dépendances des applications en matière d'accès aux données stockées dans d'autres AWS services. Nous vous recommandons de tester avec vos propres applications et environnements avant le déploiement en production afin de vérifier que votre configuration réseau répond aux exigences de vos charges de travail.

Nœud et module sur site CIDRs

Identifiez le nœud et le pod CIDRs que vous utiliserez pour vos nœuds hybrides et les charges de travail qui y sont exécutées. Le nœud CIDR est alloué depuis votre réseau sur site et le nœud CIDR est alloué depuis votre interface réseau de conteneurs (CNI) si vous utilisez un réseau superposé pour votre CNI. Vous transmettez votre nœud CIDRs et votre pod locaux CIDRs comme entrées lorsque vous créez votre cluster EKS avec les `RemotePodNetwork` champs `RemoteNodeNetwork`

et. Votre nœud local CIDRs doit être routable sur votre réseau local. Consultez la section suivante pour plus d'informations sur la routabilité du module CIDR sur site.

Les blocs CIDR du nœud et du pod locaux doivent répondre aux exigences suivantes :

1. Être dans l'une des plages `IPv4` RFC-1918 suivantes : `10.0.0.0/8``172.16.0.0/12`, ou. `192.168.0.0/16`
2. Le CIDR VPC de votre cluster EKS ou le CIDR de votre service Kubernetes ne se chevauchent pas. `IPv4`

Routage réseau du pod sur site

Lorsque vous utilisez des nœuds hybrides EKS, nous vous recommandons généralement de rendre votre module sur site CIDRs routable sur votre réseau local afin de permettre une communication et des fonctionnalités complètes entre les environnements cloud et sur site.

Réseaux de pods routables

Si vous parvenez à rendre votre réseau de modules routable sur votre réseau local, suivez les instructions ci-dessous.

1. Configurez le `RemotePodNetwork` champ pour votre cluster EKS avec le CIDR de votre pod local, vos tables de routage VPC avec le CIDR de votre pod local et le groupe de sécurité de votre cluster EKS avec le CIDR de votre pod local.
2. Vous pouvez utiliser plusieurs techniques pour rendre votre pod CIDR local routable sur votre réseau local, notamment le Border Gateway Protocol (BGP), les itinéraires statiques ou d'autres solutions de routage personnalisées. Le BGP est la solution recommandée car elle est plus évolutive et plus facile à gérer que les solutions alternatives qui nécessitent une configuration de routage personnalisée ou manuelle. AWS prend en charge les fonctionnalités BGP de Cilium et Calico pour les modules publicitaires CIDRs, voir the section called "Configurer CNI" et the section called "Pod à distance routable CIDRs" pour plus d'informations.
3. Les webhooks peuvent fonctionner sur des nœuds hybrides car le plan de contrôle EKS est capable de communiquer avec les adresses IP des pods attribuées aux webhooks.
4. Les charges de travail exécutées sur des nœuds cloud peuvent communiquer directement avec les charges de travail exécutées sur des nœuds hybrides du même cluster EKS.
5. D'autres AWS services, tels que les équilibreurs de charge des AWS applications et Amazon Managed Service for Prometheus, sont capables de communiquer avec les charges de travail

exécutées sur des nœuds hybrides afin d'équilibrer le trafic réseau et de récupérer les métriques des pods.

Réseaux de pods non routables

Si vous ne parvenez pas à rendre vos réseaux de modules routables sur votre réseau local, suivez les instructions ci-dessous.

1. Les webhooks ne peuvent pas s'exécuter sur des nœuds hybrides car ils nécessitent une connectivité entre le plan de contrôle EKS et les adresses IP des pods attribuées aux webhooks. Dans ce cas, nous vous recommandons d'exécuter des webhooks sur les nœuds cloud du même cluster EKS que vos nœuds hybrides, voir the section called "Configurer les webhooks" pour plus d'informations.
2. Les charges de travail exécutées sur des nœuds cloud ne peuvent pas communiquer directement avec les charges de travail exécutées sur des nœuds hybrides lorsque vous utilisez le VPC CNI pour les nœuds cloud et Cilium ou Calico pour les nœuds hybrides.
3. Utilisez la distribution du trafic de service pour maintenir le trafic local dans la zone d'où il provient. Pour plus d'informations sur la distribution du trafic de service, consultezthe section called "Configuration de la distribution du trafic de service".
4. Configurez votre CNI pour utiliser une mascarade de sortie ou une traduction d'adresses réseau (NAT) pour le trafic des pods lorsqu'il quitte vos hôtes locaux. Ceci est activé par défaut dans Cilium. Calico doit `natOutgoing` être réglé sur. `true`
5. D'autres AWS services, tels que les équilibreurs de charge d' AWS application et Amazon Managed Service for Prometheus, ne sont pas en mesure de communiquer avec les charges de travail exécutées sur des nœuds hybrides.

Accès requis lors de l'installation et de la mise à niveau du nœud hybride

Vous devez avoir accès aux domaines suivants pendant le processus d'installation au cours duquel vous installez les dépendances des nœuds hybrides sur vos hôtes. Ce processus peut être effectué une seule fois lorsque vous créez les images de votre système d'exploitation ou sur chaque hôte lors de l'exécution. Cela inclut l'installation initiale et la mise à niveau de la version Kubernetes de vos nœuds hybrides.

Composant	URL	Protocole	Port
Artefacts du nœud EKS (S3)	https://hybrid-assets.eks.amazonaws.com	HTTPS	443
Points de terminaison du service EKS	https://eks.*region*.amazonaws.com	HTTPS	443
Points de terminaison du service ECR	https://api.ecr.*region*.amazonaws.com	HTTPS	443
Points de terminaison EKS ECR	Voir the section called "Afficher les registres d'images Amazon" pour les points de terminaison régionaux.	HTTPS	443
Point de terminaison binaire SSM 1	https://amazon-ssm - *region* .s3.*region*.amazonaws.com	HTTPS	443
Point de terminaison 1 du service SSM	https://ssm.*region*.amazonaws.com	HTTPS	443
Point de terminaison binaire IAM Anywhere 2	https://rolesanywhere.amazonaws.com	HTTPS	443
Point de terminaison 2 du service IAM Anywhere	https://rolesanywhere.*region*.amazonaws.com	HTTPS	443

> **Note**
>
> [1] L'accès aux points de terminaison AWS SSM n'est requis que si vous utilisez des activations hybrides AWS SSM pour votre fournisseur d'informations d'identification IAM sur site.
> [2] L'accès aux points de terminaison AWS IAM n'est requis que si vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification IAM sur site.

Accès requis pour les opérations continues du cluster

L'accès réseau suivant pour votre pare-feu local est requis pour les opérations de cluster en cours.

> **Important**
>
> En fonction de votre choix de CNI, vous devez configurer des règles d'accès réseau supplémentaires pour les ports CNI. Consultez la documentation Cilium et la documentation Calico pour plus de détails.

Type	Protocole	Direction	Port	Source	Destination	Utilisation
HTTPS	TCP	Sortant	443	CIDR (s) du nœud distant	Cluster EKS IPs [1]	kubelet vers le serveur d'API Kubernetes
HTTPS	TCP	Sortant	443	Remote Pod CIDR (s)	Cluster EKS IPs [1]	Du pod au serveur d'API Kubernetes
HTTPS	TCP	Sortant	443	CIDR (s) du nœud distant	Point de terminaison du service SSM	Activations hybrides SSM, actualisation des

Type	Protocole	Direction	Port	Source	Destination	Utilisation
						informations d'identification et battements de cœur du SSM toutes les 5 minutes
HTTPS	TCP	Sortant	443	CIDR (s) du nœud distant	Point de terminaison du service IAM Anywhere	Actualisation des informations d'identification d'IAM Roles Anywhere
HTTPS	TCP	Sortant	443	Remote Pod CIDR (s)	Point de terminaison régional STS	Du pod au point de terminaison STS, uniquement requis pour IRSA

Type	Protocole	Direction	Port	Source	Destination	Utilisation
HTTPS	TCP	Sortant	443	CIDR (s) du nœud distant	Point de terminaison du service Amazon EKS Auth	Nœud vers le point de terminaison Amazon EKS Auth, uniquement requis pour Amazon EKS Pod Identity
HTTPS	TCP	Entrant	10250	Cluster EKS IPs [1]	CIDR (s) du nœud distant	Serveur d'API Kubernetes vers Kubelet
HTTPS	TCP	Entrant	Ports Webhook	Cluster EKS IPs [1]	Remote Pod CIDR (s)	Serveur d'API Kubernetes pour les webhooks

Type	Protocole	Direction	Port	Source	Destination	Utilisation
HTTPS	TCP, UDP	Entrant, sortant	53	Remote Pod CIDR (s)	Remote Pod CIDR (s)	Pod vers CoreDNS. Si vous exécutez au moins une réplique de CoreDNS dans le cloud, vous devez autoriser le trafic DNS vers le VPC sur lequel CoreDNS est exécuté.
Défini par l'utilisateur	Défini par l'utilisateur	Entrant, sortant	Ports d'applications	Remote Pod CIDR (s)	Remote Pod CIDR (s)	Pod à Pod

> **Note**
>
> [1] Celui IPs du cluster EKS. Consultez la section suivante sur les interfaces réseau élastiques Amazon EKS.

Interfaces réseau Amazon EKS

Amazon EKS attache des interfaces réseau aux sous-réseaux du VPC que vous transmettez lors de la création du cluster afin de permettre la communication entre le plan de contrôle EKS et votre VPC. Les interfaces réseau créées par Amazon EKS se trouvent après la création du cluster dans

la EC2 console Amazon ou à l'aide de la AWS CLI. Les interfaces réseau d'origine sont supprimées et de nouvelles interfaces réseau sont créées lorsque des modifications sont appliquées à votre cluster EKS, telles que des mises à niveau de version de Kubernetes. Vous pouvez restreindre la plage d'adresses IP des interfaces réseau Amazon EKS en utilisant des tailles de sous-réseaux limitées pour les sous-réseaux que vous transmettez lors de la création du cluster, ce qui facilite la configuration de votre pare-feu sur site afin de permettre la inbound/outbound connectivité à cet ensemble connu et restreint de. IPs Pour contrôler les sous-réseaux dans lesquels les interfaces réseau sont créées, vous pouvez limiter le nombre de sous-réseaux que vous spécifiez lorsque vous créez un cluster ou vous pouvez mettre à jour les sous-réseaux après avoir créé le cluster.

Les interfaces réseau mises en service par Amazon EKS contiennent une description du format`Amazon EKS` *`your-cluster-name`* . Consultez l'exemple ci-dessous pour une commande AWS CLI que vous pouvez utiliser pour trouver les adresses IP des interfaces réseau approvisionnées par Amazon EKS. `VPC_ID`Remplacez-le par l'ID du VPC que vous transmettez lors de la création du cluster.

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
 EKS))].PrivateIpAddress'
```

AWS Configuration du VPC et du sous-réseau

Les exigences existantes en matière de VPC et de sous-réseau pour Amazon EKS s'appliquent aux clusters dotés de nœuds hybrides. En outre, votre VPC CIDR ne peut pas chevaucher votre nœud et votre pod locaux. CIDRs Vous devez configurer les itinéraires dans votre table de routage VPC pour votre nœud local et éventuellement pour votre pod. CIDRs Ces itinéraires doivent être configurés pour acheminer le trafic vers la passerelle que vous utilisez pour votre connectivité réseau hybride, qui est généralement une passerelle privée virtuelle (VGW) ou une passerelle de transit (TGW). Si vous utilisez TGW ou VGW pour connecter votre VPC à votre environnement sur site, vous devez créer une pièce jointe TGW ou VGW pour votre VPC. Votre VPC doit prend en charge le nom d'hôte DNS et la résolution DNS.

Les étapes suivantes utilisent la AWS CLI. Vous pouvez également créer ces ressources dans AWS Management Console ou avec d'autres interfaces telles que AWS CloudFormation AWS CDK ou Terraform.

Étape 1 : créer un VPC

1. Exécutez la commande suivante pour créer un VPC. `VPC_CIDR`Remplacez-le par une `IPv4` plage CIDR RFC-1918 (privée) ou non RFC-1918 (publique) (par exemple). `10.0.0.0/16` Remarque : La résolution DNS, qui est une exigence d'EKS, est activée par défaut pour le VPC.

```
aws ec2 create-vpc --cidr-block VPC_CIDR
```

2. Activez les noms d'hôte DNS pour votre VPC. Remarque : la résolution DNS est activée par défaut pour le VPC. `VPC_ID`Remplacez-le par l'ID du VPC que vous avez créé à l'étape précédente.

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

Étape 2 : Création de sous-réseaux

Créez au moins 2 sous-réseaux. Amazon EKS utilise ces sous-réseaux pour les interfaces réseau du cluster. Pour plus d'informations, consultez les exigences et considérations relatives aux sous-réseaux.

1. Vous pouvez trouver les zones de disponibilité d'une AWS région à l'aide de la commande suivante. Remplacez `us-west-2` par votre région.

```
aws ec2 describe-availability-zones \
     --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. Créez un sous-réseau. Remplacez `VPC_ID` par l'ID du VPC. `SUBNET_CIDR`Remplacez-le par le bloc CIDR de votre sous-réseau (par exemple 10.0.1.0/24). Remplacez `AZ` par la zone de disponibilité dans laquelle le sous-réseau sera créé (par exemple us-west-2a). Les sous-réseaux que vous créez doivent se trouver dans au moins 2 zones de disponibilité différentes.

```
aws ec2 create-subnet \
    --vpc-id VPC_ID \
    --cidr-block SUBNET_CIDR \
    --availability-zone AZ
```

(Facultatif) Étape 3 : associer un VPC à la passerelle privée virtuelle Amazon VPC Transit Gateway (TGW) ou AWS Direct Connect (VGW)

Si vous utilisez un TGW ou un VGW, attachez votre VPC au TGW ou au VGW. Pour plus d'informations, consultez les pièces jointes Amazon VPC dans Amazon VPC Transit Gateways ou les associations de passerelles privées AWS virtuelles Direct Connect.

Passerelle de transit

Exécutez la commande suivante pour connecter un Transit Gateway. Remplacez `VPC_ID` par l'ID du VPC. Remplacez `SUBNET_ID1` et `SUBNET_ID2` par IDs les sous-réseaux que vous avez créés à l'étape précédente. `TGW_ID`Remplacez-le par l'identifiant de votre TGW.

```
aws ec2 create-transit-gateway-vpc-attachment \
    --vpc-id VPC_ID \
    --subnet-ids SUBNET_ID1 SUBNET_ID2 \
    --transit-gateway-id TGW_ID
```

Passerelle privée virtuelle

Exécutez la commande suivante pour connecter un Transit Gateway. `VPN_ID`Remplacez-le par l'ID de votre VGW. Remplacez `VPC_ID` par l'ID du VPC.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id VPN_ID \
    --vpc-id VPC_ID
```

(Facultatif) Étape 4 : Création d'une table de routage

Vous pouvez modifier la table de routage principale du VPC ou créer une table de routage personnalisée. Les étapes suivantes créent une table de routage personnalisée avec les itinéraires vers le nœud et le pod CIDRs locaux. Pour plus d'informations, consultez la section Tables de routage de sous-réseaux. Remplacez `VPC_ID` par l'ID du VPC.

```
aws ec2 create-route-table --vpc-id VPC_ID
```

Étape 5 : créer des itinéraires pour les nœuds et les pods locaux

Créez des itinéraires dans la table de routage pour chacun de vos nœuds distants locaux. Vous pouvez modifier la table de routage principale du VPC ou utiliser la table de routage personnalisée que vous avez créée à l'étape précédente.

Les exemples ci-dessous montrent comment créer des itinéraires pour votre nœud et votre pod CIDRs locaux. Dans les exemples, une passerelle de transit (TGW) est utilisée pour connecter le VPC à l'environnement sur site. Si vous disposez de plusieurs nœuds et pods sur site CIDRs, répétez les étapes pour chaque CIDR.

- Si vous utilisez une passerelle Internet ou une passerelle privée virtuelle (VGW), remplacez par`--transit-gateway-id`. `--gateway-id`
- `RT_ID`Remplacez-le par l'ID de la table de routage que vous avez créée à l'étape précédente.
- `REMOTE_NODE_CIDR`Remplacez-le par la plage CIDR que vous utiliserez pour vos nœuds hybrides.
- `REMOTE_POD_CIDR`Remplacez-le par la plage CIDR que vous utiliserez pour les pods exécutés sur des nœuds hybrides. La plage CIDR du pod correspond à la configuration CNI (Container Networking Interface), qui utilise le plus souvent un réseau superposé sur site. Pour de plus amples informations, veuillez consulter the section called “Configurer CNI”.
- `TGW_ID`Remplacez-le par l'identifiant de votre TGW.

Réseau de nœuds distants

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_NODE_CIDR \
    --transit-gateway-id TGW_ID
```

Réseau Remote Pod

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_POD_CIDR \
    --transit-gateway-id TGW_ID
```

(Facultatif) Étape 6 : associer des sous-réseaux à la table de routage

Si vous avez créé une table de routage personnalisée à l'étape précédente, associez chacun des sous-réseaux que vous avez créés à l'étape précédente à votre table de routage personnalisée. Si vous modifiez la table de routage principale du VPC, les sous-réseaux sont automatiquement associés à la table de routage principale du VPC et vous pouvez ignorer cette étape.

Exécutez la commande suivante pour chacun des sous-réseaux que vous avez créés au cours des étapes précédentes. `RT_ID`Remplacez-le par la table de routage que vous avez créée à l'étape précédente. Remplacez `SUBNET_ID` par l'ID d'un sous-réseau.

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

Configuration du groupe de sécurité du cluster

L'accès suivant pour votre groupe de sécurité de cluster EKS est requis pour les opérations de cluster en cours. Amazon EKS crée automatiquement les règles de groupe de sécurité d'entrée requises pour les nœuds hybrides lorsque vous créez ou mettez à jour votre cluster avec des réseaux de nœuds et de pods distants configurés.

Type	Protocole	Direction	Port	Source	Destination	Utilisation
HTTPS	TCP	Entrant	443	CIDR (s) du nœud distant	N/A	Serveur d'API Kubelet vers Kubernetes
HTTPS	TCP	Entrant	443	Remote Pod CIDR (s)	N/A	Pods nécessitant un accès au serveur API K8s lorsque le CNI n'utilise pas le NAT

Type	Protocole	Direction	Port	Source	Destination	Utilisation
						pour le trafic des pods.
HTTPS	TCP	Sortant	10250	N/A	CIDR (s) du nœud distant	Serveur d'API Kubernetes pour Kubelet
HTTPS	TCP	Sortant	Ports Webhook	N/A	Remote Pod CIDR (s)	Serveur d'API Kubernetes vers webhook (si vous exécutez des webhooks sur des nœuds hybrides)

Important

Limites des règles des groupes de EC2 sécurité : les groupes de sécurité Amazon ont un maximum de 60 règles d'entrée par défaut. Les règles d'entrée du groupe de sécurité peuvent ne pas s'appliquer si le groupe de sécurité de votre cluster approche cette limite. Dans ce cas, il peut être nécessaire d'ajouter manuellement les règles d'entrée manquantes. Responsabilité du nettoyage CIDR : si vous supprimez des réseaux de nœuds ou de pods distants des clusters EKS, EKS ne supprime pas automatiquement les règles de groupe de sécurité correspondantes. Il vous incombe de supprimer manuellement les réseaux de nœuds ou de modules distants non utilisés des règles de votre groupe de sécurité.

Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité".

(Facultatif) Configuration manuelle du groupe de sécurité

Si vous devez créer des groupes de sécurité supplémentaires ou modifier les règles créées automatiquement, vous pouvez utiliser les commandes suivantes comme référence. Par défaut, la commande ci-dessous crée un groupe de sécurité qui autorise tous les accès sortants. Vous pouvez restreindre l'accès sortant pour inclure uniquement les règles ci-dessus. Si vous envisagez de limiter les règles de sortie, nous vous recommandons de tester minutieusement toutes vos applications et la connectivité des pods avant d'appliquer les règles modifiées à un cluster de production.

- Dans la première commande, remplacez `SG_NAME` par le nom de votre groupe de sécurité
- Dans la première commande, remplacez `VPC_ID` par l'ID du VPC que vous avez créé à l'étape précédente
- Dans la deuxième commande, remplacez `SG_ID` par l'ID du groupe de sécurité que vous avez créé dans la première commande
- Dans la deuxième commande, remplacez `REMOTE_NODE_CIDR` et par `REMOTE_POD_CIDR` les valeurs de vos nœuds hybrides et de votre réseau sur site.

```
aws ec2 create-security-group \
    --group-name SG_NAME \
    --description "security group for hybrid nodes" \
    --vpc-id VPC_ID
```

```
aws ec2 authorize-security-group-ingress \
    --group-id SG_ID \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort": 443,
 "IpRanges": [{"CidrIp": "REMOTE_NODE_CIDR"}, {"CidrIp": "REMOTE_POD_CIDR"}]}]'
```

Préparer le système d'exploitation pour les nœuds hybrides

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu et RHEL sont validés en permanence pour être utilisés comme système d'exploitation de nœuds pour les nœuds hybrides. Bottlerocket est uniquement pris en charge dans les environnements AWS vSphere VMware . AL2Le 023 n'est pas couvert par les plans de AWS Support lorsqu'il est exécuté en dehors d'Amazon EC2. AL2Le 023 ne

peut être utilisé que dans des environnements virtualisés sur site. Consultez le guide de l'utilisateur Amazon Linux 2023 pour plus d'informations. AWS prend en charge l'intégration des nœuds hybrides avec les systèmes d'exploitation Ubuntu et RHEL, mais ne fournit pas de support pour le système d'exploitation lui-même.

Vous êtes responsable du provisionnement et de la gestion du système d'exploitation. Lorsque vous testez des nœuds hybrides pour la première fois, il est plus simple d'exécuter la CLI Amazon EKS Hybrid Nodes (`nodeadm`) sur un hôte déjà provisionné. Pour les déploiements de production, nous vous recommandons d'inclure `nodeadm` dans votre système d'exploitation des images configurées pour s'exécuter en tant que service systemd afin de joindre automatiquement les hôtes aux clusters Amazon EKS au démarrage de l'hôte. Si vous utilisez Bottlerocket comme système d'exploitation de nœud sur vSphere, vous n'avez pas besoin de l'utiliser `nodeadm` car Bottlerocket contient déjà les dépendances requises pour les nœuds hybrides et se connecte automatiquement au cluster que vous configurez au démarrage de l'hôte.

Compatibilité des versions

Le tableau ci-dessous présente les versions de système d'exploitation compatibles et validées pour être utilisées comme système d'exploitation de nœud pour les nœuds hybrides. Si vous utilisez d'autres variantes ou versions de système d'exploitation qui ne sont pas incluses dans ce tableau, la compatibilité des nœuds hybrides avec la variante ou la version de votre système d'exploitation n'est pas couverte par le AWS Support. Les nœuds hybrides sont indépendants de l'infrastructure sous-jacente et prennent en charge les architectures x86 et ARM.

Système d'exploitation	Versions
Amazon Linux	Amazon Linux 2023 (AL2023)
Bottlerocket	Variantes v1.37.0 et supérieures exécutant Kubernetes v1.28 et VMware versions ultérieures
Ubuntu	Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04
Utilisation de Red Hat Enterprise Linux	RHEL 8, RHEL 9

Considérations relatives au système d'exploitation

Général

- La CLI Amazon EKS Hybrid Nodes (`nodeadm`) peut être utilisée pour simplifier l'installation et la configuration des composants et des dépendances des nœuds hybrides. Vous pouvez exécuter le `nodeadm install` processus pendant les pipelines de génération d'images de votre système d'exploitation ou lors de l'exécution sur chaque hôte local. Pour plus d'informations sur les composants `nodeadm` installés, consultez lethe section called "Nœuds hybrides nodeadm".
- Si vous utilisez un proxy dans votre environnement local pour accéder à Internet, une configuration de système d'exploitation supplémentaire est requise pour les processus d'installation et de mise à niveau afin de configurer votre gestionnaire de packages afin qu'il utilise le proxy. Pour obtenir des instructions, consultez the section called "Configurer le proxy".

Bottlerocket

- Les étapes et les outils de connexion à un nœud Bottlerocket sont différents de ceux des autres systèmes d'exploitation et sont décrits séparément dans la sectionthe section called "Connectez des nœuds hybrides avec Bottlerocket", plutôt que dans les étapes à suivre. the section called "Connectez des nœuds hybrides"
- Les étapes de Bottlerocket n'utilisent pas l'outil CLI des nœuds hybrides,. `nodeadm`
- Seules les VMware variantes de la version v1.37.0 et supérieures de Bottlerocket sont prises en charge avec les nœuds hybrides EKS. VMware des variantes de Bottlerocket sont disponibles pour les versions v1.28 et supérieures de Kubernetes. Les autres variantes de Bottlerocket ne sont pas prises en charge en tant que système d'exploitation de nœuds hybrides. REMARQUE : les VMware variantes de Bottlerocket ne sont disponibles que pour l'architecture x86_64.

Conteneurs

- Containerd est le runtime de conteneur standard de Kubernetes et dépend des nœuds hybrides, ainsi que de tous les types de calcul des nœuds Amazon EKS. La CLI Amazon EKS Hybrid Nodes (`nodeadm`) tente d'installer des containerd pendant le `nodeadm install` processus. Vous pouvez configurer l'installation containerd au moment de l'`nodeadm install`exécution à l'aide de l'option de ligne de `--containerd-source` commande. Les options valides sont `nonedistro`, et`docker`. Si vous utilisez RHEL, cette option n'`distro`est pas valide et vous pouvez soit configurer `nodeadm` pour installer le build containerd à partir des dépôts de Docker, soit installer manuellement containerd. Lorsque vous utilisez AL2 023 ou Ubuntu, installez `nodeadm` par défaut

containerd à partir de la distribution du système d'exploitation. Si vous ne voulez pas que nodeadm installe containerd, utilisez l'option. `--containerd-source none`

Ubuntu

- Si vous utilisez Ubuntu 20.04, vous devez utiliser les activations hybrides de AWS Systems Manager comme fournisseur d'identifiants. AWS IAM Roles Anywhere n'est pas pris en charge sur Ubuntu 20.04.
- Si vous utilisez Ubuntu 24.04, vous devrez peut-être mettre à jour votre version de containerd ou modifier AppArmor votre configuration pour adopter un correctif permettant aux pods de s'arrêter correctement, voir Ubuntu #2065423. Un redémarrage est nécessaire pour appliquer les modifications au AppArmor profil. La dernière version d'Ubuntu 24.04 dispose d'une version containerd mise à jour dans son gestionnaire de paquets avec le correctif (version 1.7.19+ de containerd).

RHEL

- Si vous utilisez RHEL 8, vous devez utiliser les activations hybrides de AWS Systems Manager comme fournisseur d'identifiants. AWS IAM Roles Anywhere n'est pas pris en charge sur RHEL 8.

ARM

- Si vous utilisez du matériel ARM, un processeur compatible ARMv8 .2 avec l'extension de cryptographie (ARMv8.2+crypto) est requis pour exécuter les versions 1.31 et supérieures du module complémentaire EKS kube-proxy. Tous les systèmes Raspberry Pi antérieurs au Raspberry Pi 5, ainsi que les processeurs basés sur le Cortex-A72, ne répondent pas à cette exigence. Pour contourner le problème, vous pouvez continuer à utiliser la version 1.30 du module complémentaire EKS kube-proxy jusqu'à la fin du support étendu en juillet 2026, voir eks/latest/userguide/kubernetes -versions.html [calendrier des versions de Kubernetes, type="documentation "] ou utiliser une image kube-proxy personnalisée en amont.
- Le message d'erreur suivant dans le journal kube-proxy indique cette incompatibilité :

```
Fatal glibc error: This version of Amazon Linux requires a newer ARM64 processor
 compliant with at least ARM architecture 8.2-a with Cryptographic extensions. On EC2
 this is Graviton 2 or later.
```

Création d'images du système d'exploitation

Amazon EKS fournit des exemples de modèles de packer que vous pouvez utiliser pour créer des images de système d'exploitation qui l'incluent `nodeadm` et le configurent pour qu'il s'exécute au démarrage de l'hôte. Ce processus est recommandé pour éviter d'extraire les dépendances des nœuds hybrides individuellement sur chaque hôte et pour automatiser le processus de démarrage des nœuds hybrides. Vous pouvez utiliser les exemples de modèles Packer avec une image ISO Ubuntu 22.04, Ubuntu 24.04, RHEL 8 ou RHEL 9 et générer des images aux formats OVA, Qcow2 ou raw.

Prérequis

Avant d'utiliser les exemples de modèles Packer, les éléments suivants doivent être installés sur la machine à partir de laquelle vous exécutez Packer.

- Packer version 1.11.0 ou supérieure. Pour obtenir des instructions sur l'installation de Packer, voir Installer Packer dans la documentation de Packer.
- En cas de compilation OVAs, VMware vSphere plugin 1.4.0 ou version ultérieure
- Si vous créez des images `Qcow2` ou des images brutes, version 1.x du plugin QEMU

Définir les variables d'environnement

Avant d'exécuter la version de Packer, définissez les variables d'environnement suivantes sur la machine à partir de laquelle vous exécutez Packer.

Général

Les variables d'environnement suivantes doivent être définies pour créer des images avec tous les systèmes d'exploitation et formats de sortie.

Variable d'environnement	Type	Description
PKR_SSH_PASSWORD	Chaîne	Packer utilise les `ssh_password` variables `ssh_username` et pour se connecter en SSH à la machine créée lors du provisionnement. Cela doit correspondre aux mots de passe utilisés

Variable d'environnement	Type	Description
		pour créer l'utilisateur initial dans les fichiers Kickstart ou de données utilisateur du système d'exploitation concerné. La valeur par défaut est définie comme « builder » ou « ubuntu » selon le système d'exploit ation. Lorsque vous définisse z votre mot de passe, assurez-vous de le modifier dans le fichier correspondant `ks.cfg` ou dans `user-data` le fichier correspondant.
URL ISO	Chaîne	URL de l'ISO à utiliser. Il peut s'agir d'un lien Web à télécharger depuis un serveur ou d'un chemin absolu vers un fichier local
ISO_CHECKSUM	Chaîne	Somme de contrôle associée pour l'ISO fournie.
FOURNISSEUR D'INFORMA TIONS D'IDENTIFICATION	Chaîne	Fournisseur d'informations d'identification pour les nœuds hybrides. Les valeurs valides sont `ssm` (par défaut) pour les activations hybrides SSM et `iam` pour IAM Roles Anywhere

Variable d'environnement	Type	Description
VERSION K8S	Chaîne	Version Kubernetes pour les nœuds hybrides (par exemple). `1.31` Pour connaître les versions de Kubernetes prises en charge, consultez le fichier -versions .html eks/latest/userguide/ kubernetes [Versions prises en charge par Amazon EKS, type="documentation "].
NODEADM_ARCH	Chaîne	Architecture pour`nodeadm install`. Sélectionnez `amd` ou `arm`.

RHEL

Si vous utilisez RHEL, les variables d'environnement suivantes doivent être définies.

Variable d'environnement	Type	Description
RH_NOM D'UTILISATEUR	Chaîne	Nom d'utilisateur du gestionnaire d'abonnements RHEL
RH_PASSWORD	Chaîne	Mot de passe du gestionnaire d'abonnements RHEL
VERSION RHEL	Chaîne	Version ISO de Rhel utilisée. Les valeurs valides sont 8 ou 9.

Ubuntu

Aucune variable d'environnement spécifique à Ubuntu n'est requise.

vSphere

Si vous créez un VMware vSphere OVA, les variables d'environnement suivantes doivent être définies.

Variable d'environnement	Type	Description
VSPHERE_SERVER	Chaîne	Adresse du serveur vSphere
VSPHERE_USER	Chaîne	Nom d'utilisateur vSphere
VSPHERE_PASSWORD	Chaîne	Mot de passe vSphere
VSPHERE_DATACENTER	Chaîne	Nom du centre de données vSphere
VSPHERE_CLUSTER	Chaîne	Nom du cluster vSphere
VSPHERE_DATASTORE	Chaîne	Nom de la banque de données vSphere
VSPHERE_NETWORK	Chaîne	Nom du réseau vSphere
VSPHERE_OUTPUT_FOLDER	Chaîne	Dossier de sortie vSphere pour les modèles

QEMU

Variable d'environnement	Type	Description
FORMAT_SORTIE_PACKER	Chaîne	Format de sortie pour le générateur QEMU. Les valeurs valides sont `qcow2` et `raw`.

Valider le modèle

Avant d'exécuter votre build, validez votre modèle à l'aide de la commande suivante après avoir défini vos variables d'environnement. Remplacez-le `template.pkr.hcl` si vous utilisez un autre nom pour votre modèle.

```
packer validate template.pkr.hcl
```

Créez des images

Créez vos images à l'aide des commandes suivantes et utilisez l'`-only`indicateur pour spécifier la cible et le système d'exploitation de vos images. Remplacez-le `template.pkr.hcl` si vous utilisez un autre nom pour votre modèle.

vSphere OVAs

> **Note**
>
> Si vous utilisez RHEL avec vSphere, vous devez convertir les fichiers Kickstart en image OEMDRV et les transmettre sous forme d'ISO à partir de laquelle démarrer. Pour plus d'informations, consultez le fichier Packer Readme dans le référentiel EKS Hybrid Nodes GitHub .

Ubuntu 22.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

RHEL 8 OVA

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

RHEL 9 OVA

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

> **Note**
>
> Si vous créez une image pour un processeur hôte spécifique qui ne correspond pas à l'hôte de votre générateur, consultez la documentation de QEMU pour connaître le nom correspondant à votre processeur hôte et utilisez l'`-cpu`indicateur avec le nom du processeur hôte lorsque vous exécutez les commandes suivantes.

Ubuntu 22.04 Qcow2/Raw

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 Qcow2/Raw

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

RHEL 8 Qcow2/Raw

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

RHEL 9 Qcow2/Raw

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

Transmettre la configuration de nodeadm via les données utilisateur

Vous pouvez transmettre la configuration de vos données utilisateur via cloud-init pour `nodeadm` configurer et connecter automatiquement des nœuds hybrides à votre cluster EKS au démarrage de l'hôte. Vous trouverez ci-dessous un exemple de la manière d'y parvenir lorsque vous utilisez VMware vSphere comme infrastructure pour vos nœuds hybrides.

1. Installez la `govc` CLI en suivant les instructions du fichier readme govc on. GitHub
2. Après avoir exécuté la version Packer dans la section précédente et configuré votre modèle, vous pouvez cloner votre modèle pour créer plusieurs nœuds différents en utilisant ce qui suit. Vous devez cloner le modèle pour chaque nouvelle machine virtuelle que vous créez et qui sera utilisée pour les nœuds hybrides. Remplacez les variables de la commande ci-dessous par les valeurs de

votre environnement. La `VM_NAME` commande ci-dessous est utilisée comme vôtre `NODE_NAME` lorsque vous injectez les noms de votre VMs via votre `metadata.yaml` fichier.

```
govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
    -on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
```

3. Après avoir cloné le modèle pour chacun de vos nouveaux modèles VMs, créez un `userdata.yaml` et `metadata.yaml` pour votre VMs. VMs Vous pouvez les partager `userdata.yaml` `metadata.yaml` et vous les remplirez par machine virtuelle en suivant les étapes ci-dessous. La `nodeadm` configuration est créée et définie dans la `write_files` section de votre`userdata.yaml`. L'exemple ci-dessous utilise les activations hybrides AWS SSM comme fournisseur d'informations d'identification sur site pour les nœuds hybrides. Pour plus d'informations sur `nodeadm` la configuration, consultez lethe section called "Nœuds hybrides nodeadm".

   userdata.yaml :

```
#cloud-config
users:
  - name: # username for login. Use 'builder' for RHEL or 'ubuntu' for Ubuntu.
    passwd: # password to login. Default is 'builder' for RHEL.
    groups: [adm, cdrom, dip, plugdev, lxd, sudo]
    lock-passwd: false
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash

write_files:
  - path: /usr/local/bin/nodeConfig.yaml
    permissions: '0644'
    content: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
          cluster:
              name: # Cluster Name
              region: # AWS region
          hybrid:
              ssm:
                  activationCode: # Your ssm activation code
                  activationId: # Your ssm activation id

runcmd:
```

```
  - /usr/local/bin/nodeadm init -c file:///usr/local/bin/nodeConfig.yaml >> /var/log/
nodeadm-init.log 2>&1
```

métadonnées .yaml :

Créez un `metadata.yaml` pour votre environnement. Conservez le format de `"$NODE_NAME"` variable dans le fichier car celui-ci sera rempli de valeurs lors d'une étape ultérieure.

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
  version: 2
  ethernets:
    nics:
      match:
        name: ens*
      dhcp4: yes
```

4. Ajoutez les `metadata.yaml` fichiers `userdata.yaml` et sous forme de `gzip+base64` chaînes à l'aide des commandes suivantes. Les commandes suivantes doivent être exécutées pour chacune des commandes VMs que vous créez. `VM_NAME`Remplacez-le par le nom de la machine virtuelle que vous mettez à jour.

```
export NODE_NAME="VM_NAME"
export USER_DATA=$(gzip -c9 <userdata.yaml | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata="${USER_DATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata.encoding=gzip+base64

envsubst '$NODE_NAME' < metadata.yaml > metadata.yaml.tmp
export METADATA=$(gzip -c9 <metadata.yaml.tmp | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata="${METADATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata.encoding=gzip+base64
```

5. Allumez votre nouveau VMs, qui devrait se connecter automatiquement au cluster EKS que vous avez configuré.

```
govc vm.power -on "${NODE_NAME}"
```

Préparer les informations d'identification pour les nœuds hybrides

Les nœuds hybrides Amazon EKS utilisent des informations d'identification IAM temporaires fournies par des activations hybrides AWS SSM ou par AWS IAM Roles Anywhere pour s'authentifier auprès du cluster Amazon EKS. Vous devez utiliser des activations hybrides AWS SSM ou des rôles AWS IAM Anywhere avec la CLI Amazon EKS Hybrid Nodes (). `nodeadm` Vous ne devez pas utiliser à la fois les activations hybrides AWS SSM et les rôles AWS IAM Anywhere. Nous vous recommandons d'utiliser les activations hybrides AWS SSM si vous ne possédez pas d'infrastructure à clé publique (PKI) existante dotée d'une autorité de certification (CA) et de certificats pour vos environnements sur site. Si vous disposez d'une infrastructure PKI et de certificats sur site, utilisez AWS IAM Roles Anywhere.

Rôle IAM des nœuds hybrides

Avant de pouvoir connecter des nœuds hybrides à votre cluster Amazon EKS, vous devez créer un rôle IAM qui sera utilisé avec les activations hybrides AWS SSM ou AWS IAM Roles Anywhere pour les informations d'identification de vos nœuds hybrides. Après la création du cluster, vous utiliserez ce rôle avec une entrée d'accès Amazon EKS ou `aws-auth` ConfigMap une entrée pour associer le rôle IAM au contrôle d'accès basé sur les rôles (RBAC) de Kubernetes. Pour plus d'informations sur l'association du rôle IAM Hybrid Nodes à Kubernetes RBAC, consultez. the section called "Préparer l'accès au cluster"

Le rôle IAM de Hybrid Nodes doit disposer des autorisations suivantes.

- Autorisations `nodeadm` permettant d'utiliser l'`eks:DescribeCluster`action pour recueillir des informations sur le cluster utilisé pour connecter les nœuds hybrides au cluster. Si vous n'activez pas l'`eks:DescribeCluster`action, vous devez transmettre votre point de terminaison d'API Kubernetes, votre bundle CA de cluster et votre IPv4 CIDR de service dans la configuration du nœud à laquelle vous passez `nodeadm` lorsque vous exécutez init. `nodeadm`
- Autorisations permettant au kubelet d'utiliser des images de conteneur issues d'Amazon Elastic Container Registry (Amazon ECR), conformément à la politique d'Amazon. EC2 ContainerRegistryPullOnly

- Si vous utilisez AWS SSM, autorisations permettant à `nodeadm` init d'utiliser les activations hybrides AWS SSM, telles que définies dans la aws-managed-policy politique/.html. latest/reference/AmazonSSMManagedInstanceCore
- Si vous utilisez AWS SSM, autorisations d'utilisation de l'`ssm:DeregisterManagedInstance`action et de l'`ssm:DescribeInstanceInformation`action pour `nodeadm uninstall` désenregistrer les instances.
- (Facultatif) Autorisations permettant à l'agent d'identité du pod Amazon EKS d'utiliser l'action `eks-auth:AssumeRoleForPodIdentity` pour récupérer les informations d'identification pour les pods.

Configuration des activations hybrides AWS SSM

Avant de configurer les activations hybrides AWS SSM, vous devez avoir créé et configuré un rôle IAM de nœuds hybrides. Pour de plus amples informations, veuillez consulter the section called "Création du rôle IAM Hybrid Nodes". Suivez les instructions de la section Créer une activation hybride pour enregistrer des nœuds auprès de Systems Manager dans le guide de l'utilisateur de AWS Systems Manager afin de créer une activation hybride AWS SSM pour vos nœuds hybrides. Le code et l'identifiant d'activation que vous recevez sont utilisés `nodeadm` lorsque vous enregistrez vos hôtes en tant que nœuds hybrides auprès de votre cluster Amazon EKS. Vous pourrez revenir à cette étape ultérieurement après avoir créé et préparé vos clusters Amazon EKS pour les nœuds hybrides.

> **Important**
>
> Systems Manager renvoie immédiatement le code d'activation et l'ID à la console ou la fenêtre de commande, selon la méthode de création de l'activation. Copiez ces informations et stockez-les en lieu sûr. Si vous quittez la console ou fermez la fenêtre de commande, vous risquez de perdre ces informations. Si vous les perdez, vous devrez recréer une activation.

Par défaut, les activations hybrides AWS SSM sont actives pendant 24 heures. Vous pouvez également spécifier un `--expiration-date` moment où vous créez votre activation hybride au format horodatage, tel que. `2024-08-01T00:00:00` Lorsque vous utilisez AWS SSM comme fournisseur d'informations d'identification, le nom du nœud de vos nœuds hybrides n'est pas configurable et est généré automatiquement par SSM. AWS Vous pouvez consulter et gérer les instances gérées par AWS SSM dans la console AWS Systems Manager sous Fleet Manager. Vous pouvez enregistrer jusqu'à 1 000 nœuds hybrides standard par compte et par AWS région sans

frais supplémentaires. Toutefois, pour enregistrer plus de 1 000 nœuds hybrides, vous avez besoin d'activer le niveau d'instances avancées. L'utilisation du niveau d'instances avancées est payante et n'est pas incluse dans la tarification des nœuds hybrides Amazon EKS. Pour plus d'informations, consultez la section Tarification de AWS Systems Manager.

Consultez l'exemple ci-dessous pour savoir comment créer une activation hybride AWS SSM avec votre rôle IAM Hybrid Nodes. Lorsque vous utilisez des activations hybrides AWS SSM pour les informations d'identification de vos nœuds hybrides, les noms de vos nœuds hybrides auront le même format `mi-012345678abcdefgh` et les informations d'identification temporaires fournies par AWS SSM sont valides pendant 1 heure. Vous ne pouvez pas modifier le nom du nœud ou la durée des informations d'identification lorsque vous utilisez AWS SSM comme fournisseur d'informations d'identification. Les informations d'identification temporaires sont automatiquement modifiées par AWS SSM et la rotation n'a aucun impact sur l'état de vos nœuds ou applications.

Nous vous recommandons d'utiliser une activation hybride AWS SSM par cluster EKS pour définir l'`ssm:DeregisterManagedInstance`autorisation AWS SSM du rôle IAM des nœuds hybrides afin de ne pouvoir désenregistrer que les instances associées à votre activation hybride SSM. AWS Dans l'exemple de cette page, une balise avec l'ARN du cluster EKS est utilisée, qui peut être utilisée pour mapper votre activation hybride AWS SSM au cluster EKS. Vous pouvez également utiliser votre balise et votre méthode préférées pour définir la portée des autorisations AWS SSM en fonction de vos limites et exigences en matière d'autorisation. L'`REGISTRATION_LIMIT`option de la commande ci-dessous est un entier utilisé pour limiter le nombre de machines pouvant utiliser l'activation hybride AWS SSM (par exemple`10`)

```
aws ssm create-activation \
     --region AWS_REGION \
     --default-instance-name eks-hybrid-nodes \
     --description "Activation for EKS hybrid nodes" \
     --iam-role AmazonEKSHybridNodesRole \
     --tags Key=EKSClusterARN,Value=arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/
CLUSTER_NAME \
     --registration-limit REGISTRATION_LIMIT
```

Consultez les instructions de la section Créer une activation hybride pour enregistrer des nœuds auprès de Systems Manager pour plus d'informations sur les paramètres de configuration disponibles pour les activations hybrides AWS SSM.

Configurer des rôles AWS IAM n'importe où

Suivez les instructions de la section Getting started with IAM Roles Anywhere du guide de l'utilisateur d'IAM Roles Anywhere pour configurer l'ancre de confiance et le profil que vous utiliserez comme informations d'identification IAM temporaires pour votre rôle IAM Hybrid Nodes. Lorsque vous créez votre profil, vous pouvez le créer sans ajouter de rôles. Vous pouvez créer ce profil, revenir à ces étapes pour créer votre rôle Hybrid Nodes IAM, puis ajouter votre rôle à votre profil une fois celui-ci créé. Vous pouvez également AWS CloudFormation suivre les étapes décrites plus loin sur cette page pour terminer la configuration d'IAM Roles Anywhere pour les nœuds hybrides.

Lorsque vous ajoutez le rôle IAM Hybrid Nodes à votre profil, sélectionnez Accepter le nom de session du rôle personnalisé dans le panneau Nom de session du rôle personnalisé au bas de la page Modifier le profil de la console AWS IAM Roles Anywhere. Cela correspond au champ acceptRoleSessionNom de l'`CreateProfile`API. Cela vous permet de fournir un nom de nœud personnalisé pour vos nœuds hybrides dans la configuration à laquelle vous passez `nodeadm` pendant le processus de démarrage. Il est nécessaire de transmettre un nom de nœud personnalisé au cours du `nodeadm init` processus. Vous pouvez mettre à jour votre profil pour accepter un nom de session de rôle personnalisé après avoir créé votre profil.

Vous pouvez configurer la durée de validité des informations d'identification avec AWS IAM Roles Anywhere via le champ DurationSeconds de votre profil IAM Roles Anywhere. AWS La durée par défaut est de 1 heure avec un maximum de 12 heures. Le `MaxSessionDuration` paramètre de votre rôle IAM Hybrid Nodes doit être supérieur à celui `durationSeconds` de votre profil AWS IAM Roles Anywhere. Pour plus d'informations`MaxSessionDuration`, consultez la documentation de UpdateRole l'API.

Les certificats et clés par machine que vous générez à partir de votre autorité de certification (CA) doivent être placés dans le `/etc/iam/pki` répertoire de chaque nœud hybride avec les noms `server.pem` de fichier du certificat et `server.key` de la clé.

Création du rôle IAM Hybrid Nodes

Pour exécuter les étapes décrites dans cette section, le principal IAM utilisant la AWS console ou la AWS CLI doit disposer des autorisations suivantes.

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- Si vous utilisez AWS IAM Roles Anywhere

- `rolesanywhere:CreateTrustAnchor`
- `rolesanywhere:CreateProfile`
- `iam:PassRole`

AWS CloudFormation

Installez et configurez la AWS CLI, si ce n'est pas déjà fait. Consultez la section Installation ou mise à jour vers la dernière version de la AWS CLI.

Étapes pour les AWS activations hybrides SSM

La CloudFormation pile crée le rôle IAM des nœuds hybrides avec les autorisations décrites ci-dessus. Le CloudFormation modèle ne crée pas l'activation hybride AWS SSM.

1. Téléchargez le CloudFormation modèle AWS SSM pour les nœuds hybrides :

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ssm-cfn.yaml'
```

2. Créez un `cfn-ssm-parameters.json` avec les options suivantes :
   a. `ROLE_NAME`Remplacez-le par le nom de votre rôle IAM Hybrid Nodes. Par défaut, le CloudFormation modèle utilise `AmazonEKSHybridNodesRole` comme nom du rôle qu'il crée si vous ne spécifiez aucun nom.
   b. `TAG_KEY`Remplacez-la par la clé de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM associées à votre AWS activation hybride SSM. Dans le CloudFormation modèle, la `TAG_KEY` valeur par défaut est. `EKSClusterARN`
   c. `TAG_VALUE`Remplacez-la par la valeur de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM associées à votre AWS activation hybride SSM. Si vous utilisez la valeur par défaut `TAG_KEY` de`EKSClusterARN`, transmettez l'ARN de votre cluster EKS en tant que`TAG_VALUE`. Les clusters EKS ARNs ont le format `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. Déployez la CloudFormation pile. `STACK_NAME`Remplacez-le par le nom de la CloudFormation pile.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ssm-cfn.yaml \
    --parameter-overrides file://cfn-ssm-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

Étapes à suivre pour jouer AWS des rôles partout dans le monde

La CloudFormation pile crée l'ancre de confiance AWS IAM Roles Anywhere auprès de l'autorité de certification (CA) que vous configurez, crée le profil AWS IAM Roles Anywhere et crée le rôle IAM Hybrid Nodes avec les autorisations décrites précédemment.

1. Pour configurer une autorité de certification (CA)
   a. Pour utiliser une ressource d'autorité de certification AWS privée, ouvrez la console AWS Private Certificate Authority. Suivez les instructions du guide de l'utilisateur de AWS Private CA.
   b. Pour utiliser une autorité de certification externe, suivez les instructions fournies par l'autorité de certification. Vous fournirez le corps du certificat lors d'une étape ultérieure.
   c. Les certificats émis par le public CAs ne peuvent pas être utilisés comme points d'ancrage de confiance.
2. Téléchargez le CloudFormation modèle AWS IAM Roles Anywhere pour les nœuds hybrides

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ira-cfn.yaml'
```

3. Créez un `cfn-iamra-parameters.json` avec les options suivantes :

a. `ROLE_NAME`Remplacez-le par le nom de votre rôle IAM Hybrid Nodes. Par défaut, le CloudFormation modèle utilise `AmazonEKSHybridNodesRole` comme nom du rôle qu'il crée si vous ne spécifiez aucun nom.

b. `CERT_ATTRIBUTE`Remplacez-le par l'attribut de certificat par machine qui identifie de manière unique votre hôte. L'attribut de certificat que vous utilisez doit correspondre au nom du nœud que vous utilisez pour la `nodeadm` configuration lorsque vous connectez des nœuds hybrides à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm". Par défaut, le CloudFormation modèle utilise `${aws:PrincipalTag/x509Subject/CN}` as le`CERT_ATTRIBUTE`, qui correspond au champ CN de vos certificats par machine. Vous pouvez également vous faire `$(aws:PrincipalTag/x509SAN/Name/CN}` passer pour votre`CERT_ATTRIBUTE`.

c. `CA_CERT_BODY`Remplacez-le par le corps du certificat de votre autorité de certification sans sauts de ligne. Ils `CA_CERT_BODY` doivent être au format Privacy Enhanced Mail (PEM). Si vous avez un certificat CA au format PEM, supprimez les sauts de ligne et les lignes BEGIN CERTIFICATE et END CERTIFICATE avant de placer le corps du certificat CA dans votre `cfn-iamra-parameters.json` fichier.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
    "CABundleCert": "CA_CERT_BODY"
  }
}
```

4. Déployez le CloudFormation modèle. `STACK_NAME`Remplacez-le par le nom de la CloudFormation pile.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ira-cfn.yaml \
    --parameter-overrides file://cfn-iamra-parameters.json
    --capabilities CAPABILITY_NAMED_IAM
```

AWS CLI

Installez et configurez la AWS CLI, si ce n'est pas déjà fait. Consultez la section Installation ou mise à jour vers la dernière version de la AWS CLI.

Créer une politique de cluster EKS Describe

1. Créez un fichier nommé `eks-describe-cluster-policy.json` avec le contenu suivant :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Créez la politique à l'aide de la commande suivante :

```
aws iam create-policy \
    --policy-name EKSDescribeClusterPolicy \
    --policy-document file://eks-describe-cluster-policy.json
```

Étapes pour les AWS activations hybrides SSM

1. Créez un fichier nommé `eks-hybrid-ssm-policy.json` avec les contenus suivants. La politique autorise deux actions `ssm:DescribeInstanceInformation` et`ssm:DeregisterManagedInstance`. La politique restreint l'`ssm:DeregisterManagedInstance`autorisation aux instances gérées AWS SSM associées à votre activation hybride AWS SSM en fonction de la balise de ressource que vous spécifiez dans votre politique de confiance.
   a. Remplacez `AWS_REGION` par la AWS région pour votre activation hybride AWS SSM.
   b. Remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.
   c. `TAG_KEY`Remplacez-la par la clé de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM

associées à votre AWS activation hybride SSM. Dans le CloudFormation modèle, la `TAG_KEY` valeur par défaut est. `EKSClusterARN`

d. `TAG_VALUE`Remplacez-la par la valeur de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM associées à votre AWS activation hybride SSM. Si vous utilisez la valeur par défaut `TAG_KEY` de`EKSClusterARN`, transmettez l'ARN de votre cluster EKS en tant que`TAG_VALUE`. Les clusters EKS ARNs ont le format `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:managed-instance/
*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

2. Créez la politique à l'aide de la commande suivante

```
aws iam create-policy \
    --policy-name EKSHybridSSMPolicy \
    --policy-document file://eks-hybrid-ssm-policy.json
```

3. Créez un fichier nommé `eks-hybrid-ssm-trust.json`. `AWS_REGION`Remplacez-le par la AWS région de votre activation hybride AWS SSM et `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"AWS_ACCOUNT_ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:*"
            }
         }
      }
   ]
}
```

4. Créez le rôle à l'aide de la commande suivante.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. Joignez le `EKSDescribeClusterPolicy` et le `EKSHybridSSMPolicy` que vous avez créés lors des étapes précédentes. Remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
```

```
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSHybridSSMPolicy
```

6. Joignez les politiques `AmazonEC2ContainerRegistryPullOnly` et les politiques `AmazonSSMManagedInstanceCore` AWS gérées.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
```

Étapes à suivre pour jouer AWS des rôles partout dans le monde

Pour utiliser AWS IAM Roles Anywhere, vous devez configurer votre ancre de confiance AWS IAM Roles Anywhere avant de créer le rôle IAM Hybrid Nodes. Pour obtenir des instructions, consultez the section called "Configurer des rôles AWS IAM n'importe où".

1. Créez un fichier nommé `eks-hybrid-iamra-trust.json`. `TRUST_ANCHOR ARN`Remplacez-le par l'ARN de l'ancre de confiance que vous avez créée au cours des the section called "Configurer des rôles AWS IAM n'importe où" étapes. La condition énoncée dans cette politique de confiance limite la capacité d' AWS IAM Roles Anywhere à assumer le rôle Hybrid Nodes IAM pour échanger des informations d'identification IAM temporaires uniquement lorsque le nom de session du rôle correspond au CN figurant dans le certificat x509 installé sur vos nœuds hybrides. Vous pouvez également utiliser d'autres attributs de certificat pour identifier de manière unique votre nœud. L'attribut de certificat que vous utilisez dans la politique de confiance doit correspondre à celui `nodeName` que vous avez défini dans votre `nodeadm` configuration. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
```

```
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        }
    ]
}
```

2. Créez le rôle à l'aide de la commande suivante.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-iamra-trust.json
```

3. Joignez le `EKSDescribeClusterPolicy` fichier que vous avez créé lors des étapes précédentes. Remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. Joindre la politique `AmazonEC2ContainerRegistryPullOnly` AWS gérée

```
aws iam attach-role-policy \
```

```
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

AWS Management Console

Créer une politique de cluster EKS Describe

1. Ouvrez la console Amazon IAM
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Sur la page Politiques, choisissez Créer une politique.
4. Sur la page Spécifier les autorisations, dans le panneau Sélectionner un service, choisissez EKS.
   a. Filtrez les actions pour DescribeClusteret sélectionnez l'action DescribeClusterLire.
   b. Choisissez Suivant.
5. Sur la page Réviser et créer
   a. Entrez un nom de politique pour votre politique, tel que`EKSDescribeClusterPolicy`.
   b. Choisissez Create Policy (Créer une politique).

Étapes pour les AWS activations hybrides SSM

1. Ouvrez la console Amazon IAM
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Sur la page Politiques, choisissez Créer une politique.
4. Sur la page Spécifier les autorisations, dans le menu de navigation en haut à droite de l'éditeur de politiques, choisissez JSON. Collez l'extrait suivant. Remplacez `AWS_REGION` par la AWS région de votre activation hybride AWS SSM et remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte. Remplacez `TAG_KEY` et `TAG_VALUE` par la clé de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre activation hybride AWS SSM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

a. Choisissez Suivant.

5. Sur la page Réviser et créer.
   a. Entrez un nom de politique pour votre politique, tel que `EKSHybridSSMPolicy`
   b. Choisissez Create Policy (Créer une politique).
6. Dans le volet de navigation de gauche, choisissez Rôles.
7. Sur la page Rôles, choisissez Créer un rôle.
8. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Dans la section Type d'entité fiable, choisissez Politique de confiance personnalisée. Collez ce qui suit dans l'éditeur de politique de confiance personnalisée. `AWS_REGION`Remplacez-le par la AWS région de votre activation hybride AWS SSM et `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"AWS_ACCOUNT_ID"
            },
            "ArnEquals":{
```

```
                "aws:SourceArn":"arn:aws: ssm:AWS_REGION:AWS_ACCOUNT_ID:*"
            }
        }
    }
  ]
}
```

b. Choisissez Suivant.

9. Sur la page Ajouter des autorisations, associez une stratégie personnalisée ou procédez comme suit :

   a. Dans la zone Filtrer les politiques, entrez`EKSDescribeClusterPolicy`, ou le nom de la politique que vous avez créée ci-dessus. Cochez la case située à gauche du nom de votre politique dans les résultats de recherche.

   b. Dans la zone Filtrer les politiques, entrez`EKSHybridSSMPolicy`, ou le nom de la politique que vous avez créée ci-dessus. Cochez la case située à gauche du nom de votre politique dans les résultats de recherche.

   c. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEC2ContainerRegistryPullOnly`. Cochez la case située à gauche `AmazonEC2ContainerRegistryPullOnly` dans les résultats de recherche.

   d. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonSSMManagedInstanceCore`. Cochez la case située à gauche `AmazonSSMManagedInstanceCore` dans les résultats de recherche.

   e. Choisissez Suivant.

10. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSHybridNodesRole`.

   b. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Hybrid Nodes role`.

   c. Choisissez Créer un rôle.

Étapes à suivre pour jouer AWS des rôles partout dans le monde

Pour utiliser AWS IAM Roles Anywhere, vous devez configurer votre ancre de confiance AWS IAM Roles Anywhere avant de créer le rôle IAM Hybrid Nodes. Pour obtenir des instructions, consultez the section called “Configurer des rôles AWS IAM n'importe où”.

1. Ouvrez la console Amazon IAM
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Dans la section Type d'entité fiable, choisissez Politique de confiance personnalisée. Collez ce qui suit dans l'éditeur de politique de confiance personnalisée. `TRUST_ANCHOR ARN`Remplacez-le par l'ARN de l'ancre de confiance que vous avez créée au cours des the section called "Configurer des rôles AWS IAM n'importe où" étapes. La condition énoncée dans cette politique de confiance limite la capacité d' AWS IAM Roles Anywhere à assumer le rôle Hybrid Nodes IAM pour échanger des informations d'identification IAM temporaires uniquement lorsque le nom de session du rôle correspond au CN figurant dans le certificat x509 installé sur vos nœuds hybrides. Vous pouvez également utiliser d'autres attributs de certificat pour identifier de manière unique votre nœud. L'attribut de certificat que vous utilisez dans la politique de confiance doit correspondre au nom de nœud que vous avez défini dans votre configuration nodeadm. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "TRUST_ANCHOR_ARN"
                }
            }
        }
    ]
}
```

b. Choisissez Suivant.

5. Sur la page Ajouter des autorisations, associez une stratégie personnalisée ou procédez comme suit :

   a. Dans la zone Filtrer les politiques, entrez`EKSDescribeClusterPolicy`, ou le nom de la politique que vous avez créée ci-dessus. Cochez la case située à gauche du nom de votre politique dans les résultats de recherche.

   b. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEC2ContainerRegistryPullOnly`. Cochez la case située à gauche `AmazonEC2ContainerRegistryPullOnly` dans les résultats de recherche.

   c. Choisissez Suivant.

6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSHybridNodesRole`.

   b. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Hybrid Nodes role`.

   c. Choisissez Créer un rôle.

Création d'un cluster Amazon EKS avec des nœuds hybrides

Cette rubrique fournit une vue d'ensemble des options disponibles et décrit les éléments à prendre en compte lors de la création d'un cluster Amazon EKS compatible avec les nœuds hybrides. Les nœuds hybrides EKS ont le même eks/latest/userguide/kubernetes fichier -versions.html [support des

versions Kubernetes, type="documentation "] que les clusters Amazon EKS dotés de nœuds cloud, y compris le support standard et étendu.

Si vous ne prévoyez pas d'utiliser les nœuds hybrides EKS, consultez la documentation principale d'Amazon EKS sur la création de clusters à l'adressethe section called “Créer un cluster ”.

Prérequis

- Le the section called “Prérequis” terminé. Avant de créer votre cluster compatible avec les nœuds hybrides, vous devez CIDRs identifier votre nœud local et éventuellement votre pod, créer votre VPC et vos sous-réseaux conformément aux exigences d'EKS et aux exigences relatives aux nœuds hybrides, ainsi que votre groupe de sécurité avec des règles d'entrée pour votre espace sur site et éventuellement pour votre espace. CIDRs Pour plus d'informations sur ces conditions préalables, consultezthe section called “Préparer la mise en réseau”.
- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Les gestionnaires de packages tels que yum, apt-get ou Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la CLI. AWS Pour installer la dernière version, reportez-vous aux sections Installation ou mise à jour de la dernière version de la AWS CLI et Configuration des paramètres de la AWS CLI dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
- Un directeur IAM autorisé à créer des rôles IAM et à associer des politiques, ainsi qu'à créer et à décrire des clusters EKS

Considérations

- Votre cluster doit utiliser l'un `API` ou l'autre mode d'authentification du cluster ou `API_AND_CONFIG_MAP` pour celui-ci.
- Votre cluster doit utiliser une famille IPv4 d'adresses.
- Votre cluster doit utiliser une connectivité de point de terminaison de cluster publique ou privée. Votre cluster ne peut pas utiliser la connectivité de point de terminaison de cluster « public et privé », car le point de terminaison du serveur d'API Amazon EKS Kubernetes sera transmis au public IPs pour les nœuds hybrides exécutés en dehors de votre VPC.
- L'authentification OIDC est prise en charge pour les clusters EKS dotés de nœuds hybrides.
- Vous pouvez ajouter, modifier ou supprimer la configuration des nœuds hybrides d'un cluster existant. Pour de plus amples informations, veuillez consulter the section called “Cluster existant”.

Étape 1 : créer le rôle IAM du cluster

Si vous possédez déjà un rôle IAM dans le cluster, ou si vous allez créer votre cluster avec `eksctl` ou AWS CloudFormation, vous pouvez ignorer cette étape. Par défaut, `eksctl` le AWS CloudFormation modèle crée le rôle IAM du cluster pour vous.

1. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. Créez le rôle IAM de cluster Amazon EKS. Si nécessaire, préfacez eks-cluster-role-trust -policy.json avec le chemin sur l'ordinateur où vous avez écrit le fichier à l'étape précédente. La commande associe la politique d'approbation que vous avez créée lors de l'étape précédente à ce rôle. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role \
    --role-name myAmazonEKSClusterRole \
    --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

3. Vous pouvez attribuer la politique gérée par Amazon EKS ou créer votre propre politique personnalisée. Pour connaître les autorisations minimales que vous devez utiliser dans votre politique personnalisée, consultez the section called "Rôle IAM de nœud". Attachez la politique gérée par Amazon EKS appelée `AmazonEKSClusterPolicy` au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy \
```

```
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
    --role-name myAmazonEKSClusterRole
```

Étape 2 : créer un cluster compatible avec les nœuds hybrides

Vous pouvez créer un cluster en utilisant :

- eksctl
- AWS CloudFormation
- AWS INTERFACE DE LIGNE DE COMMANDE (CLI)
- AWS Management Console

Créer un cluster compatible avec les nœuds hybrides - eksctl

Vous devez installer la dernière version de l'outil de ligne de `eksctl` commande. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.

1. Créez `cluster-config.yaml` pour définir un cluster Amazon EKS IPv4 compatible avec les nœuds hybrides. Effectuez les remplacements suivants dans votre. `cluster-config.yaml` Pour une liste complète des paramètres, consultez la documentation eksctl.
   a. Remplacez `CLUSTER_NAME` par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
   b. `AWS_REGION`Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
   c. `K8S_VERSION`Remplacez-le par n'importe quel eks/latest/userguide/kubernetes fichier - versions.html [version prise en charge par Amazon EKS, type="documentation "].
   d. `CREDS_PROVIDER`Remplacez-le par `ssm` ou `ira` en fonction du fournisseur d'informations d'identification que vous avez configuré dans les étapes pour the section called "Préparer les informations d'identification" lesquelles.
   e. `CA_BUNDLE_CERT`Remplacez-le si votre fournisseur d'informations d'identification est défini sur`ira`, ce qui utilise AWS IAM Roles Anywhere comme fournisseur d'informations d'identification. Le CA_BUNDLE_CERT est l'organisme de certification de l'autorité de certification (CA) et dépend de votre choix d'autorité de certification. Le certificat doit être au format Privacy Enhanced Mail (PEM).

f. `GATEWAY_ID`Remplacez-le par l'ID de votre passerelle privée virtuelle ou de votre passerelle de transit à associer à votre VPC.

g. `REMOTE_NODE_CIDRS`Remplacez-le par le CIDR du nœud sur site pour vos nœuds hybrides.

h. `REMOTE_POD_CIDRS`Remplacez-le par le module CIDR sur site pour les charges de travail exécutées sur des nœuds hybrides ou supprimez la ligne de votre configuration si vous n'exécutez pas de webhooks sur des nœuds hybrides. Vous devez configurer votre CNI `REMOTE_POD_CIDRS` si celui-ci n'utilise pas la traduction d'adresses réseau (NAT) ou le masquage des adresses IP des pods lorsque le trafic des pods quitte vos hôtes locaux. Vous devez configurer `REMOTE_POD_CIDRS` si vous exécutez des webhooks sur des nœuds hybrides, voir the section called "Configurer les webhooks" pour plus d'informations.

i. Les blocs CIDR de votre nœud et de votre pod locaux doivent répondre aux exigences suivantes :

   i. Se situer dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8``172.16.0.0/12`, ou. `192.168.0.0/16`

   ii. Ne pas se chevaucher, ni `VPC CIDR` pour votre cluster, ni pour votre service Kubernetes CIDR IPv4

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: CLUSTER_NAME
  region: AWS_REGION
  version: "K8S_VERSION"

remoteNetworkConfig:
  iam:
    provider: CREDS_PROVIDER # default SSM, can also be set to IRA
    # caBundleCert: CA_BUNDLE_CERT
  vpcGatewayID: GATEWAY_ID
  remoteNodeNetworks:
  - cidrs: ["REMOTE_NODE_CIDRS"]
  remotePodNetworks:
  - cidrs: ["REMOTE_POD_CIDRS"]
```

2. Exécutez la commande suivante :

```
eksctl create cluster -f cluster-config.yaml
```

L'approvisionnement de cluster dure plusieurs minutes. Pendant la création du cluster, plusieurs lignes de sortie apparaissent. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[#]  EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. Poursuivez avec the section called "Étape 3 : Mettre à jour kubeconfig".

Créez un cluster compatible avec les nœuds hybrides - AWS CloudFormation

La CloudFormation pile crée le rôle IAM du cluster EKS et un cluster EKS avec le `RemoteNodeNetwork` et `RemotePodNetwork` que vous spécifiez. Modifiez le CloudFormation modèle si vous devez personnaliser les paramètres de votre cluster EKS qui ne sont pas exposés dans le CloudFormation modèle.

1. Téléchargez le CloudFormation modèle.

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-eks-cfn.yaml'
```

2. Créez un `cfn-eks-parameters.json` et spécifiez votre configuration pour chaque valeur.
   a. `CLUSTER_NAME`: nom du cluster EKS à créer
   b. `CLUSTER_ROLE_NAME`: nom du rôle IAM du cluster EKS à créer. La valeur par défaut du modèle est « EKSCluster Rôle ».
   c. `SUBNET1_ID`: l'ID du premier sous-réseau que vous avez créé lors des étapes préalables
   d. `SUBNET2_ID`: l'ID du deuxième sous-réseau que vous avez créé lors des étapes préalables
   e. `SG_ID`: l'ID du groupe de sécurité que vous avez créé lors des étapes préalables
   f. `REMOTE_NODE_CIDRS`: le nœud CIDR sur site pour vos nœuds hybrides
   g. `REMOTE_POD_CIDRS`: le module CIDR sur site pour les charges de travail exécutées sur des nœuds hybrides. Vous devez configurer votre CNI `REMOTE_POD_CIDRS` si celui-ci n'utilise pas la traduction d'adresses réseau (NAT) ou le masquage des adresses IP des pods lorsque le trafic des pods quitte vos hôtes locaux. Vous devez configurer `REMOTE_POD_CIDRS` si vous exécutez des webhooks sur des nœuds hybrides, voir the section called "Configurer les webhooks" pour plus d'informations.
   h. Les blocs CIDR de votre nœud et de votre pod locaux doivent répondre aux exigences suivantes :

   i. Se situer dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8172.16.0.0/12`, ou. `192.168.0.0/16`
   ii. Ne vous chevauchez pas, ni avec le CIDR `VPC CIDR` de votre cluster, ni avec le CIDR de votre service Kubernetes. IPv4

i. `CLUSTER_AUTH`: le mode d'authentification du cluster pour votre cluster. Les valeurs valides sont `API` et `API_AND_CONFIG_MAP`. La valeur par défaut du modèle est`API_AND_CONFIG_MAP`.

j. `CLUSTER_ENDPOINT`: la connectivité des points de terminaison de votre cluster. Les valeurs valides sont « Public » et « Privé ». La valeur par défaut du modèle est Private, ce qui signifie que vous ne pourrez vous connecter au point de terminaison de l'API Kubernetes que depuis votre VPC.

k. `K8S_VERSION`: la version de Kubernetes à utiliser pour votre cluster. Consultez le eks/latest/userguide/kubernetes fichier -versions.html [Versions prises en charge par Amazon EKS, type="documentation "].

```
{
  "Parameters": {
    "ClusterName": "CLUSTER_NAME",
    "ClusterRoleName": "CLUSTER_ROLE_NAME",
    "SubnetId1": "SUBNET1_ID",
    "SubnetId2": "SUBNET2_ID",
    "SecurityGroupId" "SG_ID",
    "RemoteNodeCIDR": "REMOTE_NODE_CIDRS",
    "RemotePodCIDR": "REMOTE_POD_CIDRS",
    "ClusterAuthMode": "CLUSTER_AUTH",
    "ClusterEndpointConnectivity": "CLUSTER_ENDPOINT",
    "K8sVersion": "K8S_VERSION"
  }
 }
```

3. Déployez la CloudFormation pile. `STACK_NAME`Remplacez-le par le nom de la CloudFormation pile et `AWS_REGION` par AWS la région dans laquelle le cluster sera créé.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --template-file hybrid-eks-cfn.yaml \
    --parameter-overrides file://cfn-eks-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

L'approvisionnement de cluster dure plusieurs minutes. Vous pouvez vérifier l'état de votre pile à l'aide de la commande suivante. `STACK_NAME`Remplacez-le par le nom de la CloudFormation pile et `AWS_REGION` par AWS la région dans laquelle le cluster sera créé.

```
aws cloudformation describe-stacks \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --query 'Stacks[].StackStatus'
```

4. Poursuivez avec the section called "Étape 3 : Mettre à jour kubeconfig".

Créer un cluster compatible avec les nœuds hybrides - CLI AWS

1. Exécutez la commande suivante pour créer un cluster EKS compatible avec les nœuds hybrides. Avant d'exécuter la commande, remplacez ce qui suit par vos paramètres. Pour obtenir la liste complète des paramètres, consultez la the section called "Créer un cluster " documentation.
   a. `CLUSTER_NAME`: nom du cluster EKS à créer
   b. `AWS_REGION`: AWS Région dans laquelle le cluster sera créé.
   c. `K8S_VERSION`: la version de Kubernetes à utiliser pour votre cluster. Consultez les versions prises en charge par Amazon EKS.
   d. `ROLE_ARN`: le rôle de cluster Amazon EKS que vous avez configuré pour votre cluster. Consultez le rôle IAM du cluster Amazon EKS pour plus d'informations.
   e. `SUBNET1_ID`: l'ID du premier sous-réseau que vous avez créé lors des étapes préalables
   f. `SUBNET2_ID`: l'ID du deuxième sous-réseau que vous avez créé lors des étapes préalables
   g. `SG_ID`: l'ID du groupe de sécurité que vous avez créé lors des étapes préalables
   h. Vous pouvez utiliser `API` et `API_AND_CONFIG_MAP` pour le mode d'authentification d'accès à votre cluster. Dans la commande ci-dessous, le mode d'authentification de l'accès au cluster est défini sur`API_AND_CONFIG_MAP`.
   i. Vous pouvez utiliser les `endpointPrivateAccess` paramètres `endpointPublicAccess` et pour activer ou désactiver l'accès public et privé au point de terminaison du serveur d'API Kubernetes de votre cluster. Dans la commande ci-dessous, `endpointPublicAccess` il est défini sur false et `endpointPrivateAccess` défini sur true.
   j. `REMOTE_NODE_CIDRS`: le CIDR du nœud sur site pour vos nœuds hybrides.

k. `REMOTE_POD_CIDRS`(facultatif) : le module CIDR sur site pour les charges de travail exécutées sur des nœuds hybrides.

l. Les blocs CIDR de votre nœud et de votre pod locaux doivent répondre aux exigences suivantes :

   i. Se situer dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8``172.16.0.0/12`, ou. `192.168.0.0/16`

   ii. Ne vous chevauchez pas, que ce soit `VPC CIDR` pour votre cluster Amazon EKS ou pour le CIDR de votre service Kubernetes. IPv4

```
aws eks create-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --kubernetes-version K8S_VERSION \
    --role-arn ROLE_ARN \
    --resources-vpc-config
 subnetIds=SUBNET1_ID,SUBNET2_ID,securityGroupIds=SG_ID,endpointPrivateAccess=true,endpo
 \
    --access-config authenticationMode=API_AND_CONFIG_MAP \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Le provisionnement du cluster prend quelques minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante. Remplacez `CLUSTER_NAME` par le nom du cluster que vous créez et `AWS_REGION` par la AWS région dans laquelle le cluster est créé. Ne passez pas à l'étape suivante tant que le résultat renvoyé ne l'est pas`ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Poursuivez avec the section called "Étape 3 : Mettre à jour kubeconfig".

Créez un cluster compatible avec les nœuds hybrides - AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
3. Sur la page Configurer le cluster, renseignez les champs suivants :

a. Nom : nom de votre cluster. Le nom ne peut contenir que des caractères alphanumériques (en distinguant majuscules et minuscules), des traits d'union et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.

b. Rôle IAM du cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé pour permettre au plan de contrôle Kubernetes de gérer AWS les ressources en votre nom.

c. Version Kubernetes : version de Kubernetes à utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous avez besoin d'une version antérieure.

d. Politique de mise à niveau : choisissez Extended ou Standard.

   i. Étendu : cette option prend en charge la version de Kubernetes pendant 26 mois après la date de sortie. La période de support prolongée a un coût horaire supplémentaire qui commence après la fin de la période de support standard. À la fin du support étendu, votre cluster sera automatiquement mis à niveau vers la version suivante.

   ii. Standard : cette option prend en charge la version de Kubernetes pendant 14 mois après la date de sortie. Il n'y a aucun coût supplémentaire. À la fin du support standard, votre cluster sera automatiquement mis à niveau vers la version suivante.

e. Accès au cluster : choisissez d'autoriser ou de refuser l'accès à l'administrateur du cluster et sélectionnez un mode d'authentification. Les modes d'authentification suivants sont pris en charge pour les clusters compatibles avec les nœuds hybrides.

   i. API EKS : le cluster s'approvisionnera en principaux IAM authentifiés uniquement à partir de l'entrée d'accès EKS. APIs

   ii. API EKS et ConfigMap : Le cluster fournira des principes IAM authentifiés à la fois à partir de l'entrée APIs d'accès EKS et du. `aws-auth` ConfigMap

f. Chiffrement des secrets : (facultatif) choisissez d'activer le chiffrement des secrets Kubernetes à l'aide d'une clé KMS. Vous pouvez également activer cette option une fois que vous avez créé votre cluster. Avant d'activer cette fonctionnalité, assurez-vous de bien connaître les informations contenues dansthe section called “Activer le chiffrement secret”.

g. Déplacement de zone ARC : si cette option est activée, EKS enregistrera votre cluster avec le décalage de zone ARC pour vous permettre d'utiliser le décalage de zone pour déplacer le trafic d'applications hors d'une zone de zone de zone.

h. Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called “Balisage de vos ressources ”.

i. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

4. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :

   a. VPC — Choisissez un VPC existant qui répond aux exigences d'the section called "Exigences requises pour le VPC et les sous-réseaux"Amazon EKS Hybrid Nodes. Avant de choisir un VPC, nous vous recommandons de connaître toutes les exigences et considérations de View Amazon EKS concernant les exigences réseau relatives aux VPC, aux sous-réseaux et aux nœuds hybrides. Vous ne pouvez pas modifier le VPC que vous souhaitez utiliser après la création du cluster. Si aucun VPCs n'est répertorié, vous devez d'abord en créer un. Pour plus d'informations, consultez the section called "Création d'un VPC" les exigences de mise en réseau des nœuds hybrides Amazon EKS.

   b. Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Vous devez en sélectionner au moins deux.

   c. Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS. Au moins l'un des groupes de sécurité que vous spécifiez doit disposer de règles entrantes pour votre nœud local et éventuellement pour votre pod. CIDRs Consultez les exigences de mise en réseau des nœuds hybrides Amazon EKS pour plus d'informations. Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité du cluster créé par Amazon EKS, consultez la section the section called "Exigences de groupe de sécurité" Vous pouvez modifier les règles du groupe de sécurité du cluster créé par Amazon EKS.

   d. Choisissez la famille d'adresses IP du cluster : vous devez choisir IPv4 des clusters compatibles avec les nœuds hybrides.

   e. (Facultatif) Choisissez Configurer la plage d'adresses IP du service Kubernetes et spécifiez une plage de services. IPv4

   f. Choisissez Configurer les réseaux distants pour activer les nœuds hybrides et spécifiez votre nœud local et votre pod CIDRs pour les nœuds hybrides.

   g. Vous devez configurer le CIDR de votre pod distant si votre CNI n'utilise pas la traduction d'adresses réseau (NAT) ou ne masque pas les adresses IP des pods lorsque le trafic des pods quitte vos hôtes locaux. Vous devez configurer le CIDR du pod distant si vous exécutez des webhooks sur des nœuds hybrides.

h. Les blocs CIDR de votre nœud et de votre pod locaux doivent répondre aux exigences suivantes :

   i. Se situer dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8``172.16.0.0/12`, ou. `192.168.0.0/16`

   ii. Ne pas se chevaucher, ni `VPC CIDR` pour votre cluster, ni pour votre service Kubernetes CIDR IPv4

i. Pour Accès au point de terminaison de cluster, sélectionnez une option. Une fois votre cluster créé, vous pouvez modifier cette option. Pour les clusters compatibles avec les nœuds hybrides, vous devez choisir Public ou Privé. Avant de sélectionner une option autre que par défaut, assurez-vous de vous familiariser avec les options et leurs implications. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

j. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

5. (Facultatif) Sur la page Configurer l'observabilité, choisissez les options de journalisation des mesures et du plan de contrôle à activer. Par défaut, chaque type de journal est désactivé.

   a. Pour plus d'informations sur l'option Prometheus metrics, consultez. the section called "Métriques Prometheus"

   b. Pour plus d'informations sur les options de journalisation des commandes EKS, consultezthe section called "Journaux du plan de contrôle".

   c. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

6. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster.

   a. Vous pouvez choisir autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez. Les modules complémentaires Amazon EKS qui ne sont pas compatibles avec les nœuds hybrides sont marqués de la mention « Non compatible avec les nœuds hybrides » et ils sont dotés d'une règle anti-affinité qui les empêche de s'exécuter sur des nœuds hybrides. Voir Configuration des modules complémentaires pour les nœuds hybrides pour plus d'informations. Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez rechercher les modules complémentaires AWS Marketplace disponibles en saisissant du texte dans le champ de recherche. Vous pouvez également effectuer une recherche par catégorie, fournisseur ou modèle de tarification, puis choisir les modules complémentaires dans les résultats de la recherche.

b. Certains modules complémentaires, tels que CoreDNS et kube-proxy, sont installés par défaut. Si vous désactivez l'un des modules complémentaires par défaut, cela peut affecter votre capacité à exécuter des applications Kubernetes.

c. Lorsque vous avez terminé avec cette page, choisissez`Next`.

7. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez la version que vous voulez installer.

   a. Vous pourrez toujours effectuer une mise à jour vers une version ultérieure après la création du cluster. Vous pourrez mettre à jour la configuration de chaque module complémentaire après la création du cluster. Pour plus d'informations sur la configuration des modules complémentaires, consultez the section called "Mettre à jour un module complémentaire". Pour les versions complémentaires compatibles avec les nœuds hybrides, reportez-vous àthe section called "Configuration des modules complémentaires".

   b. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

8. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez Modifier. Lorsque vous êtes satisfait, choisissez Create. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné. L'approvisionnement de cluster dure plusieurs minutes.

9. Poursuivez avec the section called "Étape 3 : Mettre à jour kubeconfig".

Étape 3 : Mettre à jour kubeconfig

Si vous avez créé votre cluster à l'aide de `eksctl`, vous pouvez ignorer cette étape. Cela est dû au fait que `eksctl` a déjà terminé cette étape pour vous. Activez cette option `kubectl` pour communiquer avec votre cluster en ajoutant un nouveau contexte au fichier de `kubectl` configuration. Pour plus d'informations sur la création et la mise à jour du fichier, consultez the section called "Cluster d'accès avec kubectl".

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

L'exemple qui suit illustre un résultat.

```
Added new context arn:aws: eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/
username/.kube/config
```

Confirmez la communication avec votre cluster en exécutant la commande suivante.

```
kubectl get svc
```

L'exemple qui suit illustre un résultat.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Étape 4 : Configuration du cluster

À l'étape suivante, consultez comment the section called "Préparer l'accès au cluster" autoriser l'accès à vos nœuds hybrides pour qu'ils rejoignent votre cluster.

Activez des nœuds hybrides sur un cluster Amazon EKS existant ou modifiez la configuration

Cette rubrique fournit une vue d'ensemble des options disponibles et décrit les éléments à prendre en compte lorsque vous ajoutez, modifiez ou supprimez la configuration des nœuds hybrides pour un cluster Amazon EKS.

Pour permettre à un cluster Amazon EKS d'utiliser des nœuds hybrides, ajoutez les plages d'adresses IP CIDR de votre nœud local et éventuellement du réseau de pods dans la `RemoteNetworkConfig` configuration. EKS utilise cette liste CIDRs pour activer la connectivité entre le cluster et vos réseaux locaux. Pour obtenir la liste complète des options lors de la mise à jour de la configuration de votre cluster, consultez le UpdateClusterConfigmanuel Amazon EKS API Reference.

Vous pouvez effectuer l'une des actions suivantes sur la configuration réseau des nœuds hybrides EKS dans un cluster :

- Ajoutez une configuration réseau à distance pour activer les nœuds hybrides EKS dans un cluster existant.
- Ajoutez, modifiez ou supprimez les réseaux de nœuds distants ou les réseaux de modules distants dans un cluster existant.
- Supprimez toutes les plages CIDR du réseau de nœuds distants pour désactiver les nœuds hybrides EKS dans un cluster existant.

Prérequis

- Avant d'activer votre cluster Amazon EKS pour les nœuds hybrides, assurez-vous que votre environnement répond aux exigences décrites et détaillées sur the section called "Préparer la mise en réseau"the section called "Préparer le système d'exploitation", etthe section called "Préparer les informations d'identification". the section called "Prérequis"
- Votre cluster doit utiliser une famille IPv4 d'adresses.
- Votre cluster doit utiliser l'un `API` ou l'autre mode d'authentification du cluster ou `API_AND_CONFIG_MAP` pour celui-ci. Le processus de modification du mode d'authentification du cluster est décrit à l'adressethe section called "Mode d'authentification".
- Nous vous recommandons d'utiliser un accès au point de terminaison public ou privé pour le point de terminaison du serveur d'API Amazon EKS Kubernetes, mais pas les deux. Si vous choisissez « Public et privé », le point de terminaison du serveur d'API Amazon EKS Kubernetes indiquera toujours au public IPs les nœuds hybrides exécutés en dehors de votre VPC, ce qui peut empêcher vos nœuds hybrides de rejoindre le cluster. Le processus de modification de l'accès réseau à votre cluster est décrit à l'adressethe section called "Accès au point de terminaison de cluster".
- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Les gestionnaires de packages tels que yum, apt-get ou Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la CLI. AWS Pour installer la dernière version, reportez-vous aux sections Installation ou mise à jour vers la dernière version de la AWS CLI et Configuration des paramètres de la AWS CLI dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
- Un responsable IAM autorisé à appeler UpdateClusterConfigvotre cluster Amazon EKS.
- Mettez à jour les modules complémentaires vers des versions compatibles avec les nœuds hybrides. Pour les versions complémentaires compatibles avec les nœuds hybrides, consultezthe section called "Configuration des modules complémentaires".
- Si vous utilisez des modules complémentaires qui ne sont pas compatibles avec les nœuds hybrides, assurez-vous que le module complémentaire DaemonSetou le déploiement respecte la règle d'affinité suivante pour empêcher le déploiement sur des nœuds hybrides. Ajoutez la règle d'affinité suivante si elle n'est pas déjà présente.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
```

```
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Considérations

L'objet `remoteNetworkConfig` JSON se comporte comme suit lors d'une mise à jour :

- Toute partie existante de la configuration que vous ne spécifiez pas est inchangée. Si vous ne spécifiez aucun des `remoteNodeNetworks` ou`remotePodNetworks`, cette partie restera la même.
- Si vous modifiez les `remotePodNetworks` listes `remoteNodeNetworks` ou les listes de CIDRs, vous devez spécifier la liste complète de celles CIDRs que vous souhaitez inclure dans votre configuration finale. Lorsque vous spécifiez une modification de la liste `remoteNodeNetworks` ou de la liste `remotePodNetworks` CIDR, EKS remplace la liste d'origine lors de la mise à jour.
- Les blocs CIDR de votre nœud et de votre pod locaux doivent répondre aux exigences suivantes :
  1. Soyez dans l'une des plages de la IPv4 RFC-1918 : 10.0.0.0/8, 172.16.0.0/12 ou 192.168.0.0/16.
  2. Il ne faut pas que tous les VPC CIDRs de votre cluster Amazon EKS ou le CIDR de votre service Kubernetes se chevauchent. IPv4

Activer des nœuds hybrides sur un cluster existant

Vous pouvez activer les nœuds hybrides EKS dans un cluster existant en utilisant :

- AWS CloudFormation
- AWS INTERFACE DE LIGNE DE COMMANDE (CLI)
- AWS Management Console

Activez les nœuds hybrides EKS dans un cluster existant - AWS CloudFormation

1. Pour activer les nœuds hybrides EKS dans votre cluster, ajoutez le `RemoteNodeNetwork` et (facultatif) `RemotePodNetwork` à votre CloudFormation modèle et mettez à jour la pile. Notez qu'il `RemoteNodeNetwork` s'agit d'une liste contenant au maximum un `Cidrs` élément et d'`Cidrs`une liste de plusieurs plages d'adresses CIDR IP.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. Passez au the section called "Préparer l'accès au cluster".

Activer les nœuds hybrides EKS dans un cluster existant - AWS CLI

1. Exécutez la commande suivante `RemoteNetworkConfig` pour activer les nœuds hybrides EKS pour votre cluster EKS. Avant d'exécuter la commande, remplacez ce qui suit par vos paramètres. Pour obtenir la liste complète des paramètres, consultez le UpdateClusterConfigmanuel Amazon EKS API Reference.
   a. `CLUSTER_NAME`: nom du cluster EKS à mettre à jour.
   b. `AWS_REGION`: AWS région dans laquelle le cluster EKS est exécuté.
   c. `REMOTE_NODE_CIDRS`: le CIDR du nœud sur site pour vos nœuds hybrides.
   d. `REMOTE_POD_CIDRS`(facultatif) : le module CIDR sur site pour les charges de travail exécutées sur des nœuds hybrides.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. La mise à jour du cluster prend plusieurs minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante. `CLUSTER_NAME`Remplacez-le par le nom du cluster que vous modifiez et `AWS_REGION` par la AWS région dans laquelle le cluster est exécuté. Ne passez pas à l'étape suivante tant que le résultat renvoyé ne l'est pas`ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Passez au the section called "Préparer l'accès au cluster".

Activez les nœuds hybrides EKS dans un cluster existant - AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Choisissez l'onglet Mise en réseau, puis sélectionnez Gérer.
4. Dans le menu déroulant, sélectionnez Réseaux distants.
5. Choisissez Configurer les réseaux distants pour activer les nœuds hybrides et spécifiez votre nœud local et votre pod CIDRs pour les nœuds hybrides.
6. Choisissez Save changes (Enregistrer les modifications) pour terminer. Attendez que le statut du cluster revienne à Actif.
7. Passez au the section called "Préparer l'accès au cluster".

Mettre à jour la configuration des nœuds hybrides dans un cluster existant

Vous pouvez effectuer des modifications `remoteNetworkConfig` dans un cluster hybride existant en utilisant l'une des méthodes suivantes :

- AWS CloudFormation
- AWS INTERFACE DE LIGNE DE COMMANDE (CLI)
- AWS Management Console

Mettre à jour la configuration hybride dans un cluster existant - AWS CloudFormation

1. Mettez à jour votre CloudFormation modèle avec les nouvelles valeurs CIDR du réseau.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

> **Note**
>
> Lors de la mise à jour `RemoteNodeNetworks` des listes `RemotePodNetworks` CIDR, incluez tout CIDRs (nouveau et existant). EKS remplace la liste complète lors des mises à

jour. L'omission de ces champs dans la demande de mise à jour permet de conserver leurs configurations existantes.

2. Mettez à jour votre CloudFormation pile avec le modèle modifié et attendez que la mise à jour de la pile soit terminée.

Mettre à jour la configuration hybride dans un cluster existant - AWS CLI

1. Pour modifier le réseau distant CIDRs, exécutez la commande suivante. Remplacez les valeurs par vos paramètres :

```
aws eks update-cluster-config
--name CLUSTER_NAME
--region AWS_REGION
--remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["NEW_REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":
["NEW_REMOTE_POD_CIDRS"]}]}'
```

> **Note**
>
> Lors de la mise à jour `remoteNodeNetworks` des listes `remotePodNetworks` CIDR, incluez tout CIDRs (nouveau et existant). EKS remplace la liste complète lors des mises à jour. L'omission de ces champs dans la demande de mise à jour permet de conserver leurs configurations existantes.

2. Attendez que le statut du cluster redevienne ACTIF avant de continuer.

Mettre à jour la configuration hybride dans un cluster existant - AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Choisissez l'onglet Mise en réseau, puis sélectionnez Gérer.
4. Dans le menu déroulant, sélectionnez Réseaux distants.
5. Mettez à jour le `Remote node networks` ci-dessous et CIDRs `Remote pod networks - Optional` selon les besoins.
6. Choisissez Enregistrer les modifications et attendez que le statut du cluster redevienne actif.

Désactiver les nœuds hybrides dans un cluster existant

Vous pouvez désactiver les nœuds hybrides EKS dans un cluster existant en utilisant :

- AWS CloudFormation
- AWS INTERFACE DE LIGNE DE COMMANDE (CLI)
- AWS Management Console

Désactiver les nœuds hybrides EKS dans un cluster existant - AWS CloudFormation

1. Pour désactiver les nœuds hybrides EKS dans votre cluster, définissez `RemoteNodeNetworks` et videz `RemotePodNetworks` les baies dans votre CloudFormation modèle et mettez à jour la pile.

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

Désactiver les nœuds hybrides EKS dans un cluster existant - AWS CLI

1. Exécutez la commande suivante pour le supprimer `RemoteNetworkConfig` de votre cluster EKS. Avant d'exécuter la commande, remplacez ce qui suit par vos paramètres. Pour obtenir la liste complète des paramètres, consultez le UpdateClusterConfigmanuel Amazon EKS API Reference.
   a. `CLUSTER_NAME`: nom du cluster EKS à mettre à jour.
   b. `AWS_REGION`: AWS région dans laquelle le cluster EKS est exécuté.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
```

2. La mise à jour du cluster prend plusieurs minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante. `CLUSTER_NAME`Remplacez-le par le nom du cluster que vous modifiez et `AWS_REGION` par la AWS région dans laquelle le cluster est exécuté. Ne passez pas à l'étape suivante tant que le résultat renvoyé ne l'est pas`ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
```

```
    --query "cluster.status"
```

Désactiver les nœuds hybrides EKS dans un cluster existant - AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Choisissez l'onglet Mise en réseau, puis sélectionnez Gérer.
4. Dans le menu déroulant, sélectionnez Réseaux distants.
5. Choisissez Configurer les réseaux distants pour activer les nœuds hybrides et supprimez tous les éléments situés CIDRs sous `Remote node networks` et`Remote pod networks - Optional`.
6. Choisissez Save changes (Enregistrer les modifications) pour terminer. Attendez que le statut du cluster revienne à Actif.

Préparer l'accès au cluster pour les nœuds hybrides

Avant de connecter des nœuds hybrides à votre cluster Amazon EKS, vous devez activer votre rôle IAM de nœuds hybrides avec les autorisations Kubernetes pour rejoindre le cluster. Consultez the section called "Préparer les informations d'identification" pour plus d'informations sur la création du rôle IAM Hybrid Nodes. Amazon EKS propose deux méthodes pour associer les principaux IAM au contrôle d'accès basé sur les rôles (RBAC) de Kubernetes, aux entrées d'accès Amazon EKS et au. `aws-auth` ConfigMap Pour plus d'informations sur la gestion des accès Amazon EKS, consultezthe section called "Accès à l'API Kubernetes".

Utilisez les procédures ci-dessous pour associer votre rôle IAM Hybrid Nodes aux autorisations Kubernetes. Pour utiliser les entrées d'accès Amazon EKS, votre cluster doit avoir été créé avec les modes `API_AND_CONFIG_MAP` d'authentification `API` or. Pour utiliser le `aws-auth` ConfigMap, votre cluster doit avoir été créé avec le mode `API_AND_CONFIG_MAP` d'authentification. Le mode d'authentification `CONFIG_MAP` uniquement n'est pas pris en charge pour les clusters Amazon EKS compatibles avec les nœuds hybrides.

Utilisation des entrées d'accès Amazon EKS pour le rôle IAM des nœuds hybrides

Il existe un type d'entrée d'accès Amazon EKS pour les nœuds hybrides nommé HYBRID_LINUX qui peut être utilisé avec un rôle IAM. Avec ce type d'entrée d'accès, le nom d'utilisateur est

automatiquement défini sur system:node : {{SessionName}}. Pour plus d'informations sur la création d'entrées d'accès, consultezthe section called “Création d'entrées d'accès”.

AWS CLI

1. La dernière version de la AWS CLI doit être installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Les gestionnaires de packages tels que yum, apt-get ou Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la CLI. AWS Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
2. Créez votre entrée d'accès à l'aide de la commande suivante. Remplacez CLUSTER_NAME par le nom de votre cluster et HYBRID_NODES_ROLE_ARN par l'ARN du rôle que vous avez créé dans les étapes pour. the section called “Préparer les informations d'identification”

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \
    --principal-arn HYBRID_NODES_ROLE_ARN \
    --type HYBRID_LINUX
```

AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom de votre cluster compatible avec les nœuds hybrides.
3. Choisissez l'onglet Access.
4. Choisissez Créer une entrée d'accès.
5. Pour IAM principal, sélectionnez le rôle IAM Hybrid Nodes que vous avez créé dans les étapes pour. the section called “Préparer les informations d'identification”
6. Pour Type, sélectionnez Hybrid Linux.
7. (Facultatif) Pour Balises, attribuez des étiquettes à l'entrée d'accès. Par exemple, pour faciliter la recherche de toutes les ressources portant la même balise.
8. Choisissez Ignorer pour passer en revue et créer. Vous ne pouvez pas ajouter de politiques à l'entrée d'accès Hybrid Linux ni modifier son étendue d'accès.
9. Vérifiez la configuration de votre entrée d'accès. Si quelque chose semble incorrect, choisissez Précédent pour revenir en arrière et corriger l'erreur. Si la configuration est correcte, choisissez Créer.

Utilisation d'aws-auth ConfigMap pour le rôle IAM des nœuds hybrides

Dans les étapes suivantes, vous allez créer ou mettre à jour le `aws-auth` ConfigMap avec l'ARN du rôle IAM des nœuds hybrides que vous avez créé dans les étapes pour the section called "Préparer les informations d'identification" lesquelles.

1. Vérifiez si vous en avez déjà un `aws-auth` ConfigMap pour votre cluster. Notez que si vous utilisez un `kubeconfig` fichier spécifique, utilisez le `--kubeconfig` drapeau.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si un s'affiche `aws-auth` ConfigMap, mettez-le à jour si nécessaire.
   a. Ouvrez le ConfigMap pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez une nouvelle entrée `mapRoles` si nécessaire. `HYBRID_NODES_ROLE_ARN`Remplacez-le par l'ARN de votre rôle IAM Hybrid Nodes. Remarque : `{{SessionName}}` le format de modèle à enregistrer dans le ConfigMap. Ne le remplacez pas par d'autres valeurs.

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

   c. Enregistrez le fichier et quittez votre éditeur de texte.

3. S'il n'en existe pas `aws-auth` ConfigMap pour votre cluster, créez-le à l'aide de la commande suivante. `HYBRID_NODES_ROLE_ARN`Remplacez-le par l'ARN de votre rôle IAM Hybrid Nodes. Notez qu'il `{{SessionName}}` s'agit du format de modèle correct à enregistrer dans le ConfigMap. Ne le remplacez pas par d'autres valeurs.

```
kubectl apply -f=/dev/stdin <<-EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
```

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
EOF
```

Exécutez des charges de travail sur site sur des nœuds hybrides

Dans un cluster EKS avec des nœuds hybrides activés, vous pouvez exécuter des applications sur site et de périphérie sur votre propre infrastructure avec les mêmes clusters, fonctionnalités et outils Amazon EKS que ceux que vous utilisez dans AWS le cloud.

Les sections suivantes contiennent des step-by-step instructions relatives à l'utilisation des nœuds hybrides.

Rubriques

- Connectez des nœuds hybrides
- Connectez des nœuds hybrides avec Bottlerocket
- Mettez à niveau les nœuds hybrides pour votre cluster
- Mises à jour de sécurité des correctifs pour les nœuds hybrides
- Supprimer les nœuds hybrides

Connectez des nœuds hybrides

> **Note**
>
> Les étapes suivantes s'appliquent aux nœuds hybrides exécutant des systèmes d'exploitation compatibles, à l'exception de Bottlerocket. Pour connaître les étapes à suivre pour connecter un nœud hybride exécutant Bottlerocket, consultez. the section called "Connectez des nœuds hybrides avec Bottlerocket"

Cette rubrique décrit comment connecter des nœuds hybrides à un cluster Amazon EKS. Une fois que vos nœuds hybrides ont rejoint le cluster, ils apparaîtront avec le statut Non prêt dans la console

Amazon EKS et dans les outils compatibles avec Kubernetes tels que kubectl. Après avoir effectué les étapes de cette page, procédez the section called "Configurer CNI" à la préparation de vos nœuds hybrides pour exécuter des applications.

Prérequis

Avant de connecter des nœuds hybrides à votre cluster Amazon EKS, assurez-vous d'avoir effectué les étapes préalables.

- Vous disposez d'une connectivité réseau entre votre environnement sur site et la AWS région hébergeant votre cluster Amazon EKS. Pour plus d'informations, consultez the section called "Préparer la mise en réseau".
- Vous disposez d'un système d'exploitation compatible pour les nœuds hybrides installés sur vos hôtes locaux. Pour plus d'informations, consultez the section called "Préparer le système d'exploitation".
- Vous avez créé votre rôle Hybrid Nodes IAM et configuré votre fournisseur d'informations d'identification sur site (activations hybrides de AWS Systems Manager ou AWS IAM Roles Anywhere). Pour plus d'informations, consultez the section called "Préparer les informations d'identification".
- Vous avez créé votre cluster Amazon EKS compatible avec les nœuds hybrides. Pour plus d'informations, consultez the section called "Créer un cluster".
- Vous avez associé votre rôle IAM Hybrid Nodes aux autorisations de contrôle d'accès basé sur les rôles (RBAC) Kubernetes. Pour plus d'informations, consultez the section called "Préparer l'accès au cluster".

Étape 1 : Installation des nœuds hybrides CLI (**`nodeadm`**) sur chaque hôte local

Si vous incluez la CLI Amazon EKS Hybrid Nodes (`nodeadm`) dans les images de votre système d'exploitation prédéfinies, vous pouvez ignorer cette étape. Pour plus d'informations sur la version des nœuds hybrides de`nodeadm`, consultezthe section called "Nœuds hybrides nodeadm".

La version des nœuds hybrides de `nodeadm` est hébergée sur Amazon S3, géré par Amazon CloudFront. Pour effectuer l'installation `nodeadm` sur chaque hôte local, vous pouvez exécuter la commande suivante à partir de vos hôtes locaux.

Pour les hôtes x86_64 :

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

Pour les hôtes ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

Ajoutez une autorisation de fichier exécutable au fichier binaire téléchargé sur chaque hôte.

```
chmod +x nodeadm
```

Étape 2 : installer les dépendances des nœuds hybrides avec **`nodeadm`**

Si vous installez les dépendances des nœuds hybrides dans des images de système d'exploitation prédéfinies, vous pouvez ignorer cette étape. La `nodeadm install` commande peut être utilisée pour installer toutes les dépendances requises pour les nœuds hybrides. Les dépendances des nœuds hybrides incluent les composants containerd, kubelet, kubectl et AWS SSM ou IAM Roles Anywhere. AWS the section called "Nœuds hybrides nodeadm"Pour plus d'informations sur les composants et les emplacements de fichiers installés par`nodeadm install`. Consultez the section called "Préparer la mise en réseau" la section dédiée aux nœuds hybrides pour plus d'informations sur les domaines qui doivent être autorisés dans votre pare-feu local pour le `nodeadm install` processus.

Exécutez la commande ci-dessous pour installer les dépendances des nœuds hybrides sur votre hôte local. La commande ci-dessous doit être exécutée avec un utilisateur ayant sudo/root accès à votre hôte.

> **Important**
>
> La CLI (`nodeadm`) des nœuds hybrides doit être exécutée avec un utilisateur ayant sudo/root accès à votre hôte.

- `K8S_VERSION`Remplacez-le par la version mineure Kubernetes de votre cluster Amazon EKS, par exemple. `1.31` Consultez le eks/latest/userguide/kubernetes fichier -versions.html [Versions prises en charge par Amazon EKS, type="documentation "] pour obtenir la liste des versions de Kubernetes prises en charge.

- `CREDS_PROVIDER`Remplacez-le par le fournisseur d'informations d'identification local que vous utilisez. Les valeurs valides concernent `ssm` AWS SSM et AWS IAM `iam-ra` Roles Anywhere.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

Étape 3 : Connectez des nœuds hybrides à votre cluster

Avant de connecter vos nœuds hybrides à votre cluster, assurez-vous d'avoir autorisé l'accès requis dans votre pare-feu sur site et dans le groupe de sécurité de votre cluster pour la communication entre nœuds to/from hybrides du plan de contrôle Amazon EKS. La plupart des problèmes rencontrés à cette étape sont liés à la configuration du pare-feu, à la configuration du groupe de sécurité ou à la configuration du rôle IAM des nœuds hybrides.

> **Important**
>
> La CLI (`nodeadm`) des nœuds hybrides doit être exécutée avec un utilisateur ayant sudo/root accès à votre hôte.

1. Créez un `nodeConfig.yaml` fichier sur chaque hôte contenant les valeurs de votre déploiement. Pour une description complète des paramètres de configuration disponibles, consultezthe section called “Nœuds hybrides nodeadm”. Si votre rôle IAM Hybrid Nodes n'est pas autorisé à effectuer cette `eks:DescribeCluster` action, vous devez transmettre le point de terminaison de l'API Kubernetes, le bundle CA du cluster et le IPv4 CIDR du service Kubernetes dans la section cluster de votre. `nodeConfig.yaml`
   a. Utilisez l'`nodeConfig.yaml`exemple ci-dessous si vous utilisez des activations hybrides AWS SSM pour votre fournisseur d'informations d'identification sur site.
      i. Remplacez `CLUSTER_NAME` par le nom de votre cluster.
      ii. Remplacez `AWS_REGION` par la AWS région hébergeant votre cluster. Par exemple, `us-west-2`.
      iii. `ACTIVATION_CODE`Remplacez-le par le code d'activation que vous avez reçu lors de la création de votre activation hybride AWS SSM. Pour plus d'informations, consultez the section called “Préparer les informations d'identification”.
      iv. `ACTIVATION_ID`Remplacez-le par l'ID d'activation que vous avez reçu lors de la création de votre activation hybride AWS SSM. Vous pouvez récupérer ces informations depuis

la console AWS Systems Manager ou depuis la `aws ssm describe-activations` commande AWS CLI.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    ssm:
      activationCode: ACTIVATION_CODE
      activationId: ACTIVATION_ID
```

b. Utilisez l'`nodeConfig.yaml`exemple ci-dessous si vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification sur site.

   i. Remplacez `CLUSTER_NAME` par le nom de votre cluster.

   ii. Remplacez `AWS_REGION` par la AWS région hébergeant votre cluster. Par exemple, `us-west-2`.

   iii. `NODE_NAME`Remplacez-le par le nom de votre nœud. Le nom du nœud doit correspondre au CN du certificat sur l'hôte si vous avez configuré la politique de confiance de votre rôle IAM de nœuds hybrides avec la condition de `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"` ressource. Le `nodeName` texte que vous utilisez ne doit pas comporter plus de 64 caractères.

   iv. `TRUST_ANCHOR_ARN`Remplacez-le par l'ARN de l'ancre de confiance que vous avez configurée dans les étapes de la section Préparer les informations d'identification pour les nœuds hybrides.

   v. `PROFILE_ARN`Remplacez-le par l'ARN de l'ancre de confiance que vous avez configurée dans les étapes pourthe section called "Préparer les informations d'identification".

   vi. `ROLE_ARN`Remplacez-le par l'ARN de votre rôle IAM Hybrid Nodes.

   vii. `CERTIFICATE_PATH`Remplacez-le par le chemin sur le disque vers le certificat de votre nœud. Si vous ne le spécifiez pas, la valeur par défaut est`/etc/iam/pki/server.pem`.

   viii `KEY_PATH`Remplacez-le par le chemin sur le disque vers la clé privée de votre certificat. Si vous ne le spécifiez pas, la valeur par défaut est`/etc/iam/pki/server.key`.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
```

```
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    iamRolesAnywhere:
      nodeName: NODE_NAME
      trustAnchorArn: TRUST_ANCHOR_ARN
      profileArn: PROFILE_ARN
      roleArn: ROLE_ARN
      certificatePath: CERTIFICATE_PATH
      privateKeyPath: KEY_PATH
```

2. Exécutez la `nodeadm init` commande avec vous `nodeConfig.yaml` pour connecter vos nœuds hybrides à votre cluster Amazon EKS.

```
nodeadm init -c file://nodeConfig.yaml
```

Si la commande ci-dessus s'exécute correctement, votre nœud hybride a rejoint votre cluster Amazon EKS. Vous pouvez le vérifier dans la console Amazon EKS en accédant à l'onglet Compute de votre cluster (assurez-vous que le principal IAM est autorisé à le consulter) ou avec. `kubectl get nodes`

> **Important**
>
> Vos nœuds auront un statut`Not Ready`, ce qui est attendu et est dû à l'absence de CNI exécuté sur vos nœuds hybrides. Si vos nœuds n'ont pas rejoint le cluster, consultezthe section called “Résolution des problèmes”.

Étape 4 : Configuration d'un CNI pour les nœuds hybrides

Pour que vos nœuds hybrides soient prêts à exécuter des applications, suivez les étapes ci-dessousthe section called “Configurer CNI”.

Connectez des nœuds hybrides avec Bottlerocket

Cette rubrique décrit comment connecter des nœuds hybrides exécutant Bottlerocket à un cluster Amazon EKS. Bottlerocket est une distribution Linux open source sponsorisée et soutenue par. AWS Bottlerocket est spécialement conçu pour héberger des charges de travail de conteneurs.

Avec Bottlerocket, vous pouvez améliorer la disponibilité des déploiements conteneurisés et réduire les coûts opérationnels en automatisant les mises à jour de votre infrastructure de conteneurs. Bottlerocket inclut uniquement les logiciels essentiels pour exécuter les conteneurs, ce qui améliore l'utilisation des ressources, réduit les menaces de sécurité et réduit les frais de gestion.

Seules les VMware variantes de la version v1.37.0 et supérieures de Bottlerocket sont prises en charge avec les nœuds hybrides EKS. VMware des variantes de Bottlerocket sont disponibles pour les versions v1.28 et supérieures de Kubernetes. Les images du système d'exploitation de ces variantes incluent le kubelet, le containerd aws-iam-authenticator et d'autres logiciels requis pour les nœuds hybrides EKS. Vous pouvez configurer ces composants à l'aide d'un fichier de paramètres Bottlerocket qui inclut les données utilisateur codées en base64 pour le bootstrap Bottlerocket et les conteneurs d'administration. La configuration de ces paramètres permet à Bottlerocket d'utiliser votre fournisseur d'informations d'identification de nœuds hybrides pour authentifier les nœuds hybrides auprès de votre cluster. Une fois que vos nœuds hybrides ont rejoint le cluster, ils apparaîtront avec leur statut `Not Ready` dans la console Amazon EKS et dans les outils compatibles avec Kubernetes tels que. `kubectl` Après avoir effectué les étapes de cette page, procédez the section called "Configurer CNI" à la préparation de vos nœuds hybrides pour exécuter des applications.

Prérequis

Avant de connecter des nœuds hybrides à votre cluster Amazon EKS, assurez-vous d'avoir effectué les étapes préalables.

- Vous disposez d'une connectivité réseau entre votre environnement sur site et la AWS région hébergeant votre cluster Amazon EKS. Pour plus d'informations, consultez the section called "Préparer la mise en réseau".
- Vous avez créé votre rôle Hybrid Nodes IAM et configuré votre fournisseur d'informations d'identification sur site (activations hybrides de AWS Systems Manager ou AWS IAM Roles Anywhere). Pour plus d'informations, consultez the section called "Préparer les informations d'identification".
- Vous avez créé votre cluster Amazon EKS compatible avec les nœuds hybrides. Pour plus d'informations, consultez the section called "Créer un cluster".
- Vous avez associé votre rôle IAM Hybrid Nodes aux autorisations de contrôle d'accès basé sur les rôles (RBAC) Kubernetes. Pour plus d'informations, consultez the section called "Préparer l'accès au cluster".

Étape 1 : Création du fichier TOML des paramètres de Bottlerocket

Pour configurer Bottlerocket pour les nœuds hybrides, vous devez créer un `settings.toml` fichier avec la configuration nécessaire. Le contenu du fichier TOML varie en fonction du fournisseur d'informations d'identification que vous utilisez (SSM ou IAM Roles Anywhere). Ce fichier sera transmis sous forme de données utilisateur lors du provisionnement de l'instance Bottlerocket.

SSM

Si vous utilisez AWS Systems Manager comme fournisseur d'informations d'identification, créez un `settings.toml` fichier avec le contenu suivant :

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "ssm"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"

[settings.host-containers.control]
enabled = true
```

Remplacez les espaces réservés par les valeurs suivantes :

- `<cluster-name>`: nom de votre cluster Amazon EKS.
- `<api-server-endpoint>`: point de terminaison du serveur d'API de votre cluster.
- `<cluster-certificate-authority>`: le bundle CA codé en base64 de votre cluster.
- `<region>`: La AWS région hébergeant votre cluster, par exemple « us-east-1 ».
- `<hostname>`: le nom d'hôte de l'instance Bottlerocket, qui sera également configuré comme nom de nœud. Il peut s'agir de n'importe quelle valeur unique de votre choix, mais elle doit respecter les conventions de dénomination des objets Kubernetes. En outre, le nom d'hôte que vous utilisez ne doit pas comporter plus de 64 caractères. REMARQUE : Lorsque vous utilisez le fournisseur SSM, ce nom d'hôte et ce nom de nœud seront remplacés par l'ID de l'instance gérée (par exemple, l'`mi-*`ID) une fois l'instance enregistrée auprès de SSM.
- `<base64-encoded-admin-container-userdata>`: Le contenu codé en base64 de la configuration du conteneur d'administration Bottlerocket. L'activation du conteneur d'administration vous permet de vous connecter à votre instance Bottlerocket via SSH pour l'exploration et le débogage du système. Bien que ce paramètre ne soit pas obligatoire, nous vous recommandons de l'activer pour faciliter le dépannage. Reportez-vous à la documentation du conteneur d'administration Bottlerocket pour plus d'informations sur l'authentification auprès du conteneur d'administration. Le conteneur d'administration prend en charge les entrées utilisateur et clé SSH au format JSON, par exemple,

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: Le contenu codé en base64 de la configuration du conteneur Bottlerocket bootstrap. Reportez-vous à la documentation du conteneur Bottlerocket bootstrap pour plus d'informations sur sa configuration. Le conteneur bootstrap est chargé d'enregistrer l'instance en tant qu'instance gérée par AWS SSM et de la joindre en tant que nœud Kubernetes sur votre cluster Amazon EKS. Les données utilisateur transmises au conteneur bootstrap prennent la forme d'une invocation de commande qui accepte en entrée le code d'activation hybride SSM et l'identifiant que vous avez créés précédemment :

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-
code> --region=<region>
```

Rôles Anywhere IAM

Si vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification, créez un `settings.toml` fichier avec le contenu suivant :

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"
config = "<base64-encoded-aws-config-file>"

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "iam-ra"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

Remplacez les espaces réservés par les valeurs suivantes :

- `<cluster-name>`: nom de votre cluster Amazon EKS.
- `<api-server-endpoint>`: point de terminaison du serveur d'API de votre cluster.
- `<cluster-certificate-authority>`: le bundle CA codé en base64 de votre cluster.
- `<region>`: La AWS région hébergeant votre cluster, par exemple « us-east-1 »

- `<hostname>`: le nom d'hôte de l'instance Bottlerocket, qui sera également configuré comme nom de nœud. Il peut s'agir de n'importe quelle valeur unique de votre choix, mais elle doit respecter les conventions de dénomination des objets Kubernetes. En outre, le nom d'hôte que vous utilisez ne doit pas comporter plus de 64 caractères. REMARQUE : Lorsque vous utilisez le fournisseur IAM-RA, le nom du nœud doit correspondre au CN du certificat sur l'hôte si vous avez configuré la politique de confiance de votre rôle IAM de nœuds hybrides avec la `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"` condition de ressource.
- `<base64-encoded-aws-config-file>`: le contenu codé en base64 de votre AWS fichier de configuration. Le contenu du fichier doit être le suivant :

```
[default]
credential_process = aws_signing_helper credential-process --certificate /root/.aws/
node.crt --private-key /root/.aws/node.key --profile-arn <profile-arn> --role-arn
 <role-arn> --trust-anchor-arn <trust-anchor-arn> --role-session-name <role-session-
name>
```

- `<base64-encoded-admin-container-userdata>`: Le contenu codé en base64 de la configuration du conteneur d'administration Bottlerocket. L'activation du conteneur d'administration vous permet de vous connecter à votre instance Bottlerocket via SSH pour l'exploration et le débogage du système. Bien que ce paramètre ne soit pas obligatoire, nous vous recommandons de l'activer pour faciliter le dépannage. Reportez-vous à la documentation du conteneur d'administration Bottlerocket pour plus d'informations sur l'authentification auprès du conteneur d'administration. Le conteneur d'administration prend en charge les entrées utilisateur et clé SSH au format JSON, par exemple,

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: Le contenu codé en base64 de la configuration du conteneur Bottlerocket bootstrap. Reportez-vous à la documentation du conteneur Bottlerocket bootstrap pour plus d'informations sur sa configuration. Le conteneur bootstrap est

chargé de créer le certificat hôte IAM Roles Anywhere et les fichiers de clé privée du certificat sur l'instance. Ils seront ensuite utilisés par le `aws_signing_helper` pour obtenir des informations d'identification temporaires afin de s'authentifier auprès de votre cluster Amazon EKS. Les données utilisateur transmises au conteneur bootstrap prennent la forme d'une invocation de commande qui accepte en entrée le contenu du certificat et de la clé privée que vous avez créés précédemment :

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

Étape 2 : approvisionner la machine virtuelle Bottlerocket vSphere avec les données utilisateur

Une fois que vous avez créé le fichier TOML, transmettez-le sous forme de données utilisateur lors de la création de la machine virtuelle vSphere. N'oubliez pas que les données utilisateur doivent être configurées avant la première mise sous tension de la machine virtuelle. Vous devrez donc le fournir lors de la création de l'instance, ou si vous souhaitez créer la machine virtuelle à l'avance, la machine virtuelle doit être hors tension jusqu'à ce que vous configuriez les données utilisateur correspondantes. Par exemple, si vous utilisez la `govc` CLI :

Création d'une machine virtuelle pour la première fois

```
govc vm.create \
  -on=true \
  -c=2 \
  -m=4096 \
  -net.adapter=<network-adapter> \
  -net=<network-name> \
  -e guestinfo.userdata.encoding="base64" \
  -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
  -template=<template-name> \
  <vm-name>
```

Mettre à jour les données utilisateur d'une machine virtuelle existante

```
govc vm.create \
    -on=false \
    -c=2 \
    -m=4096 \
    -net.adapter=<network-adapter> \
    -net=<network-name> \
    -template=<template-name> \
```

```
    <vm-name>

govc vm.change
    -vm <vm-name> \
    -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
    -e guestinfo.userdata.encoding="base64"

govc vm.power -on <vm-name>
```

Dans les sections ci-dessus, l'`-e guestinfo.userdata.encoding="base64"`option indique que les données utilisateur sont codées en base64. L'`-e guestinfo.userdata`option transmet le contenu codé en base64 du `settings.toml` fichier sous forme de données utilisateur à l'instance Bottlerocket. Remplacez les espaces réservés par vos valeurs spécifiques, telles que le modèle Bottlerocket OVA et les détails du réseau.

Étape 3 : vérifier la connexion au nœud hybride

Une fois l'instance Bottlerocket démarrée, elle tentera de rejoindre votre cluster Amazon EKS. Vous pouvez vérifier la connexion dans la console Amazon EKS en accédant à l'onglet Compute de votre cluster ou en exécutant la commande suivante :

```
kubectl get nodes
```

> **Important**
>
> Vos nœuds auront un statut`Not Ready`, ce qui est attendu et est dû à l'absence de CNI exécuté sur vos nœuds hybrides. Si vos nœuds n'ont pas rejoint le cluster, consultezthe section called “Résolution des problèmes”.

Étape 4 : Configuration d'un CNI pour les nœuds hybrides

Pour que vos nœuds hybrides soient prêts à exécuter des applications, suivez les étapes ci-dessousthe section called “Configurer CNI”.

Mettez à niveau les nœuds hybrides pour votre cluster

Les instructions relatives à la mise à niveau des nœuds hybrides sont similaires à celles des nœuds Amazon EKS autogérés qui s'exécutent sur Amazon EC2. Nous vous recommandons de créer

de nouveaux nœuds hybrides sur votre version cible de Kubernetes, de migrer harmonieusement vos applications existantes vers les nœuds hybrides de la nouvelle version de Kubernetes et de supprimer les nœuds hybrides de l'ancienne version de Kubernetes de votre cluster. Assurez-vous de consulter les meilleures pratiques d'Amazon EKS pour les mises à niveau avant de lancer une mise à niveau. Les nœuds hybrides Amazon EKS disposent du même eks/latest/userguide/kubernetes fichier -versions.html [support des versions Kubernetes, type="documentation "] pour les clusters Amazon EKS dotés de nœuds cloud, y compris le support standard et étendu.

Les nœuds hybrides Amazon EKS suivent la même politique de distorsion de version pour les nœuds que pour Kubernetes en amont. Les nœuds hybrides Amazon EKS ne peuvent pas être installés sur une version plus récente que le plan de contrôle Amazon EKS, et les nœuds hybrides peuvent avoir jusqu'à trois versions mineures de Kubernetes antérieures à la version mineure du plan de contrôle Amazon EKS.

Si vous ne disposez pas de la capacité nécessaire pour créer de nouveaux nœuds hybrides sur votre version cible de Kubernetes dans le cadre d'une stratégie de migration et de mise à niveau, vous pouvez également utiliser la CLI Amazon EKS Hybrid Nodes (`nodeadm`) pour mettre à niveau la version Kubernetes de vos nœuds hybrides sur place.

> **Important**
>
> Si vous mettez à niveau vos nœuds hybrides sur place avec`nodeadm`, le nœud est indisponible pendant le processus au cours duquel l'ancienne version des composants Kubernetes est arrêtée et les nouveaux composants de la version Kubernetes sont installés et démarrés.

Prérequis

Avant de procéder à la mise à niveau, assurez-vous d'avoir rempli les conditions préalables suivantes.

- La version cible de Kubernetes pour la mise à niveau de vos nœuds hybrides doit être égale ou inférieure à la version du plan de contrôle Amazon EKS.
- Si vous suivez une stratégie de migration et de mise à niveau par transition, les nouveaux nœuds hybrides que vous installez sur votre version cible de Kubernetes doivent répondre aux exigences. the section called “Prérequis” Cela inclut le fait d'avoir des adresses IP dans le réseau de nœuds distants CIDR que vous avez transmis lors de la création du cluster Amazon EKS.

- Que ce soit pour la migration par transition ou pour les mises à niveau sur place, les nœuds hybrides doivent avoir accès aux domaines requis pour extraire les nouvelles versions des dépendances des nœuds hybrides.
- Kubectl doit être installé sur votre machine ou instance locale que vous utilisez pour interagir avec votre point de terminaison d'API Amazon EKS Kubernetes.
- La version de votre CNI doit prendre en charge la version de Kubernetes vers laquelle vous effectuez la mise à niveau. Si ce n'est pas le cas, mettez à niveau votre version CNI avant de mettre à niveau vos nœuds hybrides. Pour plus d'informations, consultez the section called "Configurer CNI".

Mises à niveau liées à la migration vers le cloud (bleu-vert)

Les mises à niveau de migration par transition font référence au processus qui consiste à créer de nouveaux nœuds hybrides sur de nouveaux hôtes avec votre version cible de Kubernetes, à migrer progressivement vos applications existantes vers les nouveaux nœuds hybrides de votre version cible de Kubernetes et à supprimer les nœuds hybrides de l'ancienne version de Kubernetes de votre cluster. Cette stratégie est également appelée migration bleu-vert.

1. Connectez vos nouveaux hôtes en tant que nœuds hybrides en suivant les the section called "Connectez des nœuds hybrides" étapes. Lorsque vous exécutez la `nodeadm install` commande, utilisez votre version cible de Kubernetes.
2. Activez la communication entre les nouveaux nœuds hybrides de la version cible de Kubernetes et vos nœuds hybrides de l'ancienne version de Kubernetes. Cette configuration permet aux pods de communiquer entre eux pendant que vous migrez votre charge de travail vers les nœuds hybrides de la version cible de Kubernetes.
3. Vérifiez que vos nœuds hybrides de votre version cible de Kubernetes ont bien rejoint votre cluster et qu'ils ont le statut Prêt.
4. Utilisez la commande suivante pour marquer comme non planifiable chacun des nœuds que vous souhaitez supprimer. Cela permet d'éviter que de nouveaux pods ne soient planifiés ou reprogrammés sur les nœuds que vous remplacez. Pour plus d'informations, consultez kubectl cordon dans la documentation de Kubernetes. `NODE_NAME`Remplacez-le par le nom des nœuds hybrides de l'ancienne version de Kubernetes.

```
kubectl cordon NODE_NAME
```

Vous pouvez identifier et boucler tous les nœuds d'une version particulière de Kubernetes (dans ce cas,`1.28`) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Cordoning $node"
    kubectl cordon $node
done
```

5. Si votre déploiement actuel exécute moins de deux répliques CoreDNS sur vos nœuds hybrides, augmentez le déploiement à au moins deux répliques. Nous vous recommandons d'exécuter au moins deux répliques CoreDNS sur des nœuds hybrides pour garantir la résilience lors des opérations normales.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

6. Videz chacun des nœuds hybrides de l'ancienne version de Kubernetes que vous souhaitez supprimer de votre cluster à l'aide de la commande suivante. Pour plus d'informations sur le drainage des nœuds, consultez la section Drainage sécurisé d'un nœud dans la documentation de Kubernetes. `NODE_NAME`Remplacez-le par le nom des nœuds hybrides de l'ancienne version de Kubernetes.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

Vous pouvez identifier et vider tous les nœuds d'une version particulière de Kubernetes (dans ce cas,`1.28`) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

7. Vous pouvez l'utiliser `nodeadm` pour arrêter et supprimer les artefacts des nœuds hybrides de l'hôte. Vous devez exécuter `nodeadm` avec un utilisateur disposant de root/sudo privilèges. Par défaut, ne `nodeadm uninstall` se poursuivra pas s'il reste des pods sur le nœud. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm".

```
nodeadm uninstall
```

8. Une fois les artefacts des nœuds hybrides arrêtés et désinstallés, supprimez la ressource du nœud de votre cluster.

```
kubectl delete node node-name
```

Vous pouvez identifier et supprimer tous les nœuds d'une version particulière de Kubernetes (dans ce cas,`1.28`) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Deleting $node"
    kubectl delete node $node
done
```

9. Selon le CNI que vous avez choisi, il se peut que des artefacts restent sur vos nœuds hybrides après avoir exécuté les étapes ci-dessus. Pour plus d'informations, consultez the section called "Configurer CNI".

Mises à niveau sur place

Le processus de mise à niveau sur place consiste à mettre `nodeadm upgrade` à niveau la version de Kubernetes pour les nœuds hybrides sans utiliser de nouveaux hôtes physiques ou virtuels et sans stratégie de migration de transition. Le `nodeadm upgrade` processus arrête les anciens composants Kubernetes existants exécutés sur le nœud hybride, désinstalle les anciens composants Kubernetes existants, installe les nouveaux composants Kubernetes cibles et démarre les nouveaux composants Kubernetes cibles. Il est vivement recommandé de mettre à niveau un nœud à la fois afin de minimiser l'impact sur les applications exécutées sur les nœuds hybrides. La durée de ce processus dépend de la bande passante et de la latence de votre réseau.

1. Utilisez la commande suivante pour marquer le nœud que vous mettez à niveau comme non planifiable. Cela permet d'éviter que de nouveaux pods ne soient planifiés ou reprogrammés sur le nœud que vous mettez à niveau. Pour plus d'informations, consultez kubectl cordon dans la documentation de Kubernetes. Remplacez `NODE_NAME` par le nom du nœud hybride que vous mettez à niveau

```
kubectl cordon NODE_NAME
```

2. Videz le nœud que vous mettez à niveau à l'aide de la commande suivante. Pour plus d'informations sur le drainage des nœuds, consultez la section Drainage sécurisé d'un nœud dans la documentation de Kubernetes. `NODE_NAME`Remplacez-le par le nom du nœud hybride que vous mettez à niveau.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

3. Exécutez `nodeadm upgrade` sur le nœud hybride que vous mettez à niveau. Vous devez exécuter `nodeadm` avec un utilisateur disposant de root/sudo privilèges. Le nom du nœud est préservé lors de la mise à niveau pour les fournisseurs d'informations d'identification AWS SSM et AWS IAM Roles Anywhere. Vous ne pouvez pas changer de fournisseur d'informations d'identification pendant le processus de mise à niveau. Voir the section called "Nœuds hybrides nodeadm" pour les valeurs de configuration pour`nodeConfig.yaml`. Remplacez-le `K8S_VERSION` par la version cible de Kubernetes vers laquelle vous effectuez la mise à niveau.

```
nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
```

4. Pour autoriser la planification des pods sur le nœud après la mise à niveau, tapez ce qui suit. Remplacez `NODE_NAME` par le nom du nœud.

```
kubectl uncordon NODE_NAME
```

5. Surveillez l'état de vos nœuds hybrides et attendez qu'ils s'arrêtent et redémarrent sur la nouvelle version de Kubernetes avec le statut Prêt.

```
kubectl get nodes -o wide -w
```

Mises à jour de sécurité des correctifs pour les nœuds hybrides

Cette rubrique décrit la procédure permettant d'appliquer des correctifs sur place aux mises à jour de sécurité pour des packages et dépendances spécifiques exécutés sur vos nœuds hybrides. En tant que bonne pratique, nous vous recommandons de mettre régulièrement à jour vos nœuds hybrides pour recevoir CVEs des correctifs de sécurité.

Pour connaître les étapes de mise à niveau de la version de Kubernetes, consultez. the section called "Mettre à niveau les nœuds hybrides"

Voici un exemple de logiciel susceptible de nécessiter des correctifs de sécurité. `containerd`

`Containerd`

`containerd`est le moteur d'exécution standard des conteneurs Kubernetes et la principale dépendance des nœuds hybrides EKS, utilisés pour gérer le cycle de vie des conteneurs, notamment l'extraction d'images et la gestion de l'exécution des conteneurs. Sur un nœud hybride, vous pouvez effectuer l'installation `containerd` via la CLI nodeadm ou manuellement. Selon le système d'exploitation de votre nœud, l'installation `nodeadm` se fera `containerd` à partir du package distribué par le système d'exploitation ou du package Docker.

Lorsqu'un fichier CVE `containerd` a été publié, vous disposez des options suivantes pour passer à la version corrigée de `containerd` sur vos nœuds hybrides.

Étape 1 : vérifier si le correctif a été publié dans les gestionnaires de packages

Vous pouvez vérifier si le correctif `containerd` CVE a été publié dans chaque gestionnaire de packages de système d'exploitation en consultant les bulletins de sécurité correspondants :

- Amazon Linux 2023
- RHEL
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Si vous utilisez le dépôt Docker comme source de`containerd`, vous pouvez consulter les annonces de sécurité Docker pour identifier la disponibilité de la version corrigée dans le référentiel Docker.

Étape 2 : Choisissez la méthode d'installation du correctif

Il existe trois méthodes pour appliquer des correctifs et installer des mises à niveau de sécurité sur place sur les nœuds. La méthode que vous pouvez utiliser dépend de la disponibilité ou non du correctif depuis le système d'exploitation dans le gestionnaire de packages :

1. Installez les correctifs `nodeadm upgrade` qui sont publiés dans les gestionnaires de packages, voir Étape 2 a.
2. Installez les correctifs directement avec les gestionnaires de packages, voir Étape 2 b.
3. Installez des correctifs personnalisés qui ne sont pas publiés dans les gestionnaires de packages. Notez qu'il existe des considérations spéciales pour les correctifs personnalisés pour`containerd`, Step 2 c.

Étape 2 a : Appliquer des correctifs avec **`nodeadm upgrade`**

Après avoir confirmé que le correctif `containerd` CVE a été publié sur le système d'exploitation ou les dépôts Docker (Apt ou RPM), vous pouvez utiliser la `nodeadm upgrade` commande pour passer à la dernière version de. `containerd` Comme il ne s'agit pas d'une mise à niveau de la version de Kubernetes, vous devez transmettre votre version actuelle de Kubernetes à la commande de mise à niveau. `nodeadm`

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

Étape 2 b : Appliquer des correctifs à l'aide des gestionnaires de packages du système d'exploitation

Vous pouvez également effectuer la mise à jour via le gestionnaire de packages correspondant et l'utiliser pour mettre à niveau le `containerd` package comme suit.

Amazon Linux 2023

```
sudo yum update -y
sudo yum install -y containerd
```

RHEL

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-
ce.repo
```

```
sudo yum update -y
sudo yum install -y containerd
```

Ubuntu

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
 https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update -y
sudo apt install -y --only-upgrade containerd.io
```

Étape 2 c : le correctif **`Containerd`** CVE n'est pas publié dans les gestionnaires de packages

Si la `containerd` version corrigée n'est disponible que par d'autres moyens plutôt que dans le gestionnaire de packages, par exemple dans les GitHub versions, vous pouvez l'installer `containerd` depuis le GitHub site officiel.

1. Si la machine a déjà rejoint le cluster en tant que nœud hybride, vous devez exécuter la `nodeadm uninstall` commande.
2. Installez les `containerd` fichiers binaires officiels. Vous pouvez suivre les étapes d'installation officielles GitHub.
3. Exécutez la `nodeadm install` commande avec l'`--containerd-source`argument défini sur`none`, ce qui annulera `containerd` l'installation`nodeadm`. Vous pouvez utiliser la valeur de `none` dans la `containerd` source pour tous les systèmes d'exploitation exécutés par le nœud.

   ```
   nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
    none
   ```

Supprimer les nœuds hybrides

Cette rubrique explique comment supprimer des nœuds hybrides de votre cluster Amazon EKS. Vous devez supprimer vos nœuds hybrides avec l'outil compatible Kubernetes de votre choix, tel que kubectl. Les frais pour les nœuds hybrides cessent lorsque l'objet du nœud est supprimé du cluster

Amazon EKS. Pour plus d'informations sur la tarification des nœuds hybrides, consultez la section Tarification d'Amazon EKS.

> **Important**
>
> La suppression de nœuds perturbe les charges de travail exécutées sur le nœud. Avant de supprimer des nœuds hybrides, nous vous recommandons de vider d'abord le nœud pour déplacer les pods vers un autre nœud actif. Pour plus d'informations sur le drainage des nœuds, consultez la section Drainage sécurisé d'un nœud dans la documentation de Kubernetes.

Exécutez les étapes kubectl ci-dessous depuis votre machine ou instance locale que vous utilisez pour interagir avec le point de terminaison de l'API Kubernetes du cluster Amazon EKS. Si vous utilisez un `kubeconfig` fichier spécifique, utilisez le `--kubeconfig` drapeau.

Étape 1 : Répertoriez vos nœuds

```
kubectl get nodes
```

Étape 2 : Videz votre nœud

Consultez kubectl drain dans la documentation de Kubernetes pour plus d'informations sur la commande. `kubectl drain`

```
kubectl drain --ignore-daemonsets <node-name>
```

Étape 3 : arrêter et désinstaller les artefacts des nœuds hybrides

Vous pouvez utiliser la CLI Amazon EKS Hybrid Nodes (`nodeadm`) pour arrêter et supprimer les artefacts des nœuds hybrides de l'hôte. Vous devez exécuter `nodeadm` avec un utilisateur qui possède root/sudo des privilèges. Par défaut, ne `nodeadm uninstall` se poursuivra pas s'il reste des pods sur le nœud. Si vous utilisez AWS Systems Manager (SSM) comme fournisseur d'informations d'identification, la `nodeadm uninstall` commande annule l'enregistrement de l'hôte en tant AWS qu'instance gérée par SSM. Pour de plus amples informations, veuillez consulter the section called “Nœuds hybrides nodeadm”.

```
nodeadm uninstall
```

Étape 4 : supprimer votre nœud du cluster

Une fois les artefacts des nœuds hybrides arrêtés et désinstallés, supprimez la ressource du nœud de votre cluster.

```
kubectl delete node <node-name>
```

Étape 5 : Vérifiez les artefacts restants

Selon le CNI que vous avez choisi, il se peut que des artefacts restent sur vos nœuds hybrides après avoir exécuté les étapes ci-dessus. Pour plus d'informations, consultez the section called "Configurer CNI".

Configurer le CNI, les modules complémentaires et les webhooks pour les nœuds hybrides

Après avoir configuré un cluster pour les nœuds hybrides, les nœuds hybrides doivent être configurés pour les interfaces réseau de conteneurs (CNI), les modules complémentaires, les webhooks et les éventuels paramètres de proxy. Pour obtenir la liste complète des modules complémentaires EKS et communautaires compatibles avec les nœuds hybrides, consultezthe section called "Configuration des modules complémentaires".

EKS Cluster Insights EKS inclut des contrôles analytiques pour détecter les erreurs de configuration dans la configuration de vos nœuds hybrides EKS susceptibles de nuire au fonctionnement de votre cluster ou à vos charges de travail. Pour plus d'informations sur les clusters, consultezthe section called "Informations sur le cluster".

La liste suivante répertorie les modules complémentaires et autres composants dont la configuration est différente pour les nœuds hybrides :

- Interface réseau de conteneurs (CNI) : les fonctionnalités de base de Cilium et Calico sont prises en charge pour une utilisation avec des nœuds hybrides. Vous pouvez gérer votre CNI sur des nœuds hybrides avec l'outillage de votre choix, tel que Helm. Le AWS VPC CNI ne peut pas être utilisé avec des nœuds hybrides. Pour de plus amples informations, veuillez consulter the section called "Configurer CNI".
- CoreDNS et **`kube-proxy`** : CoreDNS et sont installés automatiquement lorsque des nœuds hybrides rejoignent `kube-proxy` le cluster EKS. Ces modules complémentaires peuvent être gérés en tant que modules complémentaires EKS après la création du cluster.

- Ingress et équilibrage de charge : vous pouvez utiliser le AWS Load Balancer Controller et l'Application Load Balancer (ALB) ou Network Load Balancer (NLB) avec le type de `ip` cible pour les charges de travail sur des nœuds hybrides connectés à Direct Connect ou à un VPN. AWS AWS Site-to-Site Vous pouvez également utiliser le contrôleur d'entrée ou l'équilibreur de charge de votre choix pour le trafic applicatif qui reste local dans votre environnement sur site.
- Métriques : vous pouvez utiliser les scrapers sans agent Amazon Managed Service for Prometheus (AMP) AWS , Distro for Open Telemetry (ADOT) et Amazon Observability Agent avec des nœuds hybrides. CloudWatch Pour utiliser les scrapers sans agent AMP pour les métriques des pods sur des nœuds hybrides, vos pods doivent être accessibles depuis le VPC que vous utilisez pour le cluster EKS.
- Journaux : vous pouvez activer la journalisation du plan de contrôle EKS pour les clusters compatibles avec les nœuds hybrides. Vous pouvez utiliser le module complémentaire ADOT EKS et le module complémentaire Amazon CloudWatch Observability Agent EKS pour la journalisation hybride des nœuds et des pods.
- Identités de modules et IRSA : vous pouvez utiliser les identités de modules EKS et les rôles IAM pour les comptes de service (IRSA) avec des applications exécutées sur des nœuds hybrides afin de permettre un accès granulaire à vos pods exécutés sur des nœuds hybrides avec d'autres services. AWS
- Webhooks : si vous utilisez des webhooks, consultez les considérations et les étapes à suivre the section called "Configurer les webhooks" pour éventuellement exécuter des webhooks sur des nœuds cloud si vous ne pouvez pas rendre vos réseaux de pods sur site routables.
- Proxy : si vous utilisez un serveur proxy dans votre environnement sur site pour le trafic sortant de votre centre de données ou de votre environnement périphérique, vous pouvez configurer vos nœuds hybrides et votre cluster pour utiliser votre serveur proxy. Pour de plus amples informations, veuillez consulter the section called "Configurer le proxy".

Rubriques

- Configuration d'un CNI pour les nœuds hybrides
- Configuration des modules complémentaires pour les nœuds hybrides
- Configurer les webhooks pour les nœuds hybrides
- Configuration du proxy pour les nœuds hybrides

Configuration d'un CNI pour les nœuds hybrides

Cilium et Calico sont pris en charge en tant qu'interfaces réseau de conteneurs () CNIs pour les nœuds hybrides Amazon EKS. Vous devez installer un CNI pour les nœuds hybrides afin d'être prêts à prendre en charge les charges de travail. Les nœuds hybrides apparaissent avec un statut `Not Ready` jusqu'à ce qu'un CNI soit en cours d'exécution. Vous pouvez les gérer CNIs avec les outils de votre choix tels que Helm. Le CNI Amazon VPC n'est pas compatible avec les nœuds hybrides et le CNI VPC est configuré avec une anti-affinité pour l'étiquette. `eks.amazonaws.com/compute-type: hybrid`

Compatibilité des versions CNI

La version Cilium `v1.16.x` est prise en charge et recommandée pour les nœuds hybrides EKS pour chaque version de Kubernetes prise en charge dans Amazon EKS.

La version Calico `v3.29.x` est prise en charge et recommandée pour les nœuds hybrides EKS pour chaque version de Kubernetes prise en charge dans Amazon EKS.

Consultez la section Prise en charge des versions de Kubernetes pour les versions de Kubernetes prises en charge par Amazon EKS. Les nœuds hybrides EKS prennent en charge les mêmes versions de Kubernetes que les clusters Amazon EKS dotés de nœuds cloud.

Fonctionnalités prises en charge

AWS fournit un support technique pour les fonctionnalités suivantes de Cilium et Calico à utiliser avec des nœuds hybrides. Si vous envisagez d'utiliser des fonctionnalités hors du champ d' AWS assistance, nous vous recommandons d'obtenir un support commercial pour le plugin ou de disposer de l'expertise interne nécessaire pour résoudre les problèmes et apporter des correctifs au projet de plugin CNI.

Fonctionnalité	Cilium	Calico
Conformité du réseau Kubernetes	Oui	Oui
Connectivité entre le plan de contrôle et le nœud	Oui	Oui
Connectivité entre le plan de contrôle et le module	Oui	Oui

Fonctionnalité	Cilium	Calico
La gestion du cycle de vie	Installation, mise à niveau, suppression	Installation, mise à niveau, suppression
Mode réseau	VXLAN	VXLAN
Gestion des adresses IP (IPAM)	Étendue du cluster (Cilium IPAM)	Calicot IPAM
Famille d'IP	IPv4	IPv4
BGP	Oui (Cilium Control Plane)	Oui

Considérations relatives au cilium

- Par défaut, Cilium est configuré pour fonctionner en mode overlay/tunnel avec VXLAN comme méthode d'encapsulation. Ce mode présente le moins d'exigences sur le réseau physique sous-jacent.
- Par défaut, Cilium masque l'adresse IP source de tout le trafic du pod quittant le cluster sur l'adresse IP du nœud. Cela permet d'exécuter Cilium avec des nœuds hybrides, que les réseaux de pods distants soient configurés ou non sur le cluster. Si vous désactivez le masquage, votre pod CIDRs doit être routable sur votre réseau local et vous devez configurer votre cluster Amazon EKS avec vos réseaux de pods distants.
- Si vous exécutez des webhooks sur vos nœuds hybrides, votre pod CIDRs doit être routable sur votre réseau local et vous devez configurer votre cluster Amazon EKS avec vos réseaux de pods distants. Si votre pod CIDRs n'est pas routable sur votre réseau local, nous vous recommandons d'exécuter des webhooks sur les nœuds cloud du même cluster. Voir Configurer les webhooks pour les nœuds hybrides pour plus d'informations.
- Une méthode courante pour rendre le CIDR de votre pod routable sur votre réseau local consiste à publier les adresses des pods avec BGP. Pour utiliser BGP avec Cilium, vous devez le configurer `bgpControlPlane.enabled: true` dans votre configuration Helm. Pour plus d'informations sur le support BGP de Cilium, consultez le plan de contrôle BGP de Cilium dans la documentation de Cilium.
- La gestion des adresses IP (IPAM) par défaut dans Cilium s'appelle Cluster Scope, où l'opérateur Cilium alloue des adresses IP à chaque nœud en fonction du pod configuré par l'utilisateur.

CIDRs Les pods CIDRs sont configurés avec la valeur `clusterPoolIPv4PodCIDRList` Helm, qui doit correspondre au réseau de pods distants CIDRs que vous avez configuré pour votre cluster Amazon EKS. Cilium alloue des segments de à chaque nœud. `clusterPoolIPv4PodCIDRList` La taille des segments par nœud est configurée avec la valeur `clusterPoolIPv4MaskSize` Helm. Pour plus d'informations sur le `clusterPoolIPv4PodCIDRList` et`clusterPoolIPv4MaskSize`, consultez la section Extension du pool de clusters dans la documentation de Cilium.

Installez Cilium sur des nœuds hybrides

1. Assurez-vous d'avoir installé la CLI Helm sur votre environnement de ligne de commande. Consultez le guide de configuration pour les instructions d'installation.
2. Installez le dépôt Cilium Helm.

```
helm repo add cilium https://helm.cilium.io/
```

3. Créez un fichier YAML appelé`cilium-values.yaml`. L'exemple suivant configure Cilium pour qu'il s'exécute uniquement sur des nœuds hybrides en définissant l'affinité pour l'étiquette. `eks.amazonaws.com/compute-type: hybrid`
   - Si vous avez configuré votre cluster Amazon EKS avec des réseaux de modules distants, configurez le même espace CIDRs pour votre`clusterPoolIPv4PodCIDRList`. Par exemple, `10.100.0.0/24`. Le CIDR de votre pod local ne doit pas chevaucher le CIDR de votre nœud local lorsque vous exécutez le CNI en mode overlay/tunnel.
   - Configurez en `clusterPoolIPv4MaskSize` fonction des pods requis par nœud. Par exemple, 25 pour une taille de segment /25 de 128 pods par nœud.
   - Vous ne devez pas modifier votre `clusterPoolIPv4PodCIDRList` ou `clusterPoolIPv4MaskSize` après le déploiement de Cilium sur votre cluster, voir Expansion du pool de clusters dans la documentation de Cilium.
   - Pour une liste complète des valeurs Helm pour Cilium, consultez la référence Helm dans la documentation de Cilium.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
```

```
          operator: In
          values:
          - hybrid
ipam:
  mode: cluster-pool
  operator:
    clusterPoolIPv4MaskSize: 25
    clusterPoolIPv4PodCIDRList:
    - POD_CIDR
loadBalancer:
  serviceTopology: true
operator:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
              - hybrid
  unmanagedPodWatcher:
    restart: false
envoy:
  enabled: false
```

4. Installez Cilium sur votre cluster.

   - `CILIUM_VERSION`Remplacez-le par une version Cilium (par exemple`v1.17.0`). Nous vous recommandons d'exécuter la dernière version du correctif pour votre version mineure de Cilium. Vous trouverez le dernier correctif pour une version mineure donnée de Cilium dans la section Versions stables de la documentation de Cilium.
   - Si vous activez BGP pour votre déploiement, ajoutez l'`--set bgpControlPlane.enabled=true`indicateur dans la commande ci-dessous.
   - Si vous utilisez un fichier kubeconfig spécifique, utilisez l'`--kubeconfig`indicateur avec la commande Helm install.

```
helm install cilium cilium/cilium \
    --version CILIUM_VERSION \
    --namespace kube-system \
    --values cilium-values.yaml
```

5. Vous pouvez vérifier que votre installation de Cilium a réussi à l'aide des commandes suivantes. Vous devriez voir le `cilium-operator` déploiement et l'`cilium-agent`exécution sur chacun de vos nœuds hybrides. De plus, vos nœuds hybrides devraient désormais avoir un statut`Ready`. Pour plus d'informations sur la configuration de BGP pour Cilium, passez à l'étape suivante.

```
kubectl get pods -n kube-system
```

```
NAME                              READY   STATUS    RESTARTS   AGE
cilium-jjjn8                      1/1     Running   0          11m
cilium-operator-d4f4d7fcb-sc5xn   1/1     Running   0          11m
```

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES    AGE   VERSION
mi-04a2cf999b7112233   Ready    <none>   19m   v1.31.0-eks-a737599
```

6. Pour utiliser BGP avec Cilium afin de publier les adresses de vos pods sur votre réseau local, vous devez avoir installé Cilium avec. `bgpControlPlane.enabled: true` Si vous activez BGP pour un déploiement Cilium existant, vous devez redémarrer l'opérateur Cilium pour appliquer la configuration BGP si BGP n'était pas activé auparavant. Vous pouvez le `operator.rollOutPods` définir `true` dans vos valeurs Helm pour redémarrer l'opérateur Cilium dans le cadre du processus Helm install/upgrade .

   Pour configurer BGP dans Cilium, créez d'abord un fichier appelé `cilium-bgp-cluster.yaml` avec une définition. `CiliumBGPClusterConfig` Vous devrez peut-être obtenir les informations suivantes auprès de votre administrateur réseau.

   - Configurez `localASN` avec l'ASN pour les nœuds exécutant Cilium.
   - Configurez `peerASN` avec l'ASN de votre routeur local.
   - Configurez le `peerAddress` avec l'adresse IP du routeur local à laquelle chaque nœud exécutant Cilium sera homologue.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPClusterConfig
metadata:
  name: cilium-bgp
spec:
  nodeSelector:
```

```
      matchExpressions:
      - key: eks.amazonaws.com/compute-type
        operator: In
        values:
        - hybrid
    bgpInstances:
    - name: "rack0"
      localASN: NODES_ASN
      peers:
      - name: "onprem-router"
        peerASN: ONPREM_ROUTER_ASN
        peerAddress: ONPREM_ROUTER_IP
        peerConfigRef:
          name: "cilium-peer"
```

7. Appliquez la configuration du cluster Cilium BGP à votre cluster.

```
kubectl apply -f cilium-bgp-cluster.yaml
```

8. La `CiliumBGPPeerConfig` ressource définit une configuration homologue BGP. Plusieurs homologues peuvent partager la même configuration et fournir une référence à la `CiliumBGPPeerConfig` ressource commune. Créez un fichier nommé `cilium-bgp-peer.yaml` pour configurer la configuration homologue de votre réseau local. Consultez la configuration du pair BGP dans la documentation de Cilium pour une liste complète des options de configuration.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPPeerConfig
metadata:
  name: cilium-peer
spec:
  timers:
    holdTimeSeconds: 30
    keepAliveTimeSeconds: 10
  gracefulRestart:
    enabled: true
    restartTimeSeconds: 120
  families:
    - afi: ipv4
      safi: unicast
      advertisements:
        matchLabels:
```

```
          advertise: "bgp"
```

9. Appliquez la configuration Cilium BGP Peer à votre cluster.

```
kubectl apply -f cilium-bgp-peer.yaml
```

10. La `CiliumBGPAdvertisement` ressource est utilisée pour définir différents types de publicités et les attributs qui leur sont associés. Créez un fichier nommé `cilium-bgp-advertisement.yaml` et configurez la `CiliumBGPAdvertisement` ressource avec vos paramètres.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "PodCIDR"
    - advertisementType: "Service"
      service:
        addresses:
          - ClusterIP
          - ExternalIP
          - LoadBalancerIP
```

11. Appliquez la configuration de publicité Cilium BGP à votre cluster.

```
kubectl apply -f cilium-bgp-advertisement.yaml
```

Vous pouvez confirmer que le peering BGP a fonctionné avec la CLI Cilium à l'aide de la commande. `cilium bgp peers` Vous devriez voir les valeurs correctes dans la sortie pour votre environnement et l'état de la session sous la forme`established`. Consultez le guide de dépannage et d'exploitation dans la documentation de Cilium pour plus d'informations sur le dépannage.

Mettez à niveau Cilium sur des nœuds hybrides

Avant de mettre à niveau votre déploiement de Cilium, consultez attentivement la documentation de mise à niveau de Cilium et les notes de mise à niveau pour comprendre les modifications apportées à la version cible de Cilium.

1. Assurez-vous d'avoir installé la `helm` CLI sur votre environnement de ligne de commande. Consultez la documentation Helm pour les instructions d'installation.
2. Installez le dépôt Cilium Helm.

```
helm repo add cilium https://helm.cilium.io/
```

3. Exécutez le contrôle de mise à niveau de Cilium avant le vol. `CILIUM_VERSION`Remplacez-le par votre version cible de Cilium. Nous vous recommandons d'exécuter la dernière version du correctif pour votre version mineure de Cilium. Vous trouverez le dernier correctif pour une version mineure donnée de Cilium dans la section Versions stables de la documentation de Cilium.

```
helm install cilium-preflight cilium/cilium --version CILIUM_VERSION \
  --namespace=kube-system \
  --set preflight.enabled=true \
  --set agent=false \
  --set operator.enabled=false
```

4. Après l'application`cilium-preflight.yaml`, assurez-vous que le nombre de `READY` gousses est le même que le nombre de gousses de cilium en circulation.

```
kubectl get ds -n kube-system | sed -n '1p;/cilium/p'
```

```
NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
cilium                    2         2         2       2            2           <none>
          1h20m
cilium-pre-flight-check   2         2         2       2            2           <none>
          7m15s
```

5. Une fois que le nombre de pods READY est égal, assurez-vous que le déploiement de Cilium avant le vol est également marqué comme READY 1/1. S'il indique READY 0/1, consultez la section de validation CNP et résolvez les problèmes liés au déploiement avant de poursuivre la mise à niveau.

```
kubectl get deployment -n kube-system cilium-pre-flight-check -w
```

```
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
cilium-pre-flight-check   1/1     1            0           12s
```

6. Supprimer le pré-vol

```
helm uninstall cilium-preflight --namespace kube-system
```

7. Pendant le fonctionnement normal du cluster, tous les composants de Cilium doivent exécuter la même version. Les étapes suivantes décrivent comment mettre à niveau tous les composants d'une version stable vers une version stable ultérieure. Lorsque vous passez d'une version mineure à une autre, nous vous recommandons de passer d'abord au dernier correctif pour la version mineure existante de Cilium. Pour minimiser les perturbations, définissez l'`upgradeCompatibility`option sur la version initiale de Cilium que vous avez installée dans ce cluster.

   Avant d'exécuter la commande helm upgrade, conservez les valeurs de votre déploiement dans a `cilium-values.yaml` ou utilisez les options de ligne de `--set` commande pour vos paramètres. L'opération de mise à niveau remplace le Cilium ConfigMap. Il est donc essentiel que vos valeurs de configuration soient transmises lors de la mise à niveau. Si vous utilisez BGP, nous vous recommandons d'utiliser l'option de ligne de `--set bgpControlPlane=true` commande au lieu de fournir ces informations dans votre fichier de valeurs.

```
helm upgrade cilium cilium/cilium --version CILIUM_VERSION \
  --namespace kube-system \
  --set upgradeCompatibility=1.X \
  -f cilium-values.yaml
```

8. (Facultatif) Si vous devez annuler votre mise à niveau en raison de problèmes, exécutez les commandes suivantes.

```
helm history cilium --namespace kube-system
helm rollback cilium [REVISION] --namespace kube-system
```

Supprimer Cilium des nœuds hybrides

1. Exécutez la commande suivante pour désinstaller tous les composants Cilium de votre cluster. Notez que la désinstallation du CNI peut avoir un impact sur l'état des nœuds et des pods et ne doit pas être effectuée sur des clusters de production.

   ```
   helm uninstall cilium --namespace kube-system
   ```

   Les interfaces et les routes configurées par Cilium ne sont pas supprimées par défaut lorsque le CNI est supprimé du cluster. Consultez le GitHub problème pour plus d'informations.

2. Pour nettoyer les fichiers de configuration et les ressources sur disque, si vous utilisez les répertoires de configuration standard, vous pouvez supprimer les fichiers comme indiqué par le `cni-uninstall.sh`script dans le référentiel Cilium sur. GitHub

3. Pour supprimer les définitions de ressources personnalisées Cilium (CRDs) de votre cluster, vous pouvez exécuter les commandes suivantes.

   ```
   kubectl get crds -oname | grep "cilium" | xargs kubectl delete
   ```

Considérations relatives à Calico

- Nous vous recommandons d'exécuter Calico en mode overlay/tunnel avec VXLAN comme méthode d'encapsulation. Ce mode présente le moins d'exigences sur le réseau physique sous-jacent. Pour plus d'informations sur les différents modes de mise en réseau de Calico, consultez la section Déterminer la meilleure option de mise en réseau dans la documentation de Calico.
- Nous vous recommandons d'exécuter Calico avec la valeur `natOutgoing` définie sur. `true` Lorsque cette option est `natOutgoing` définie sur`true`, l'adresse IP source de tout le trafic du pod quittant le cluster est convertie en adresse IP du nœud. Cela permet d'exécuter Calico avec des clusters Amazon EKS, que des réseaux de pods distants soient configurés ou non sur le cluster. Si vous le désactive`natOutgoing`, votre pod CIDRs doit être routable sur votre réseau local et vous devez configurer votre cluster Amazon EKS avec vos réseaux de pods distants.
- Si vous exécutez des webhooks sur vos nœuds hybrides, votre pod CIDRs doit être routable sur votre réseau local et vous devez configurer votre cluster Amazon EKS avec vos réseaux de pods distants. Si votre pod CIDRs n'est pas routable sur votre réseau local, nous vous recommandons d'exécuter des webhooks sur les nœuds cloud du même cluster. Voir Configurer les webhooks pour les nœuds hybrides pour plus d'informations.

- Une méthode courante pour rendre le CIDR de votre pod routable sur votre réseau local consiste à publier les adresses des pods avec BGP. Pour utiliser BGP avec Calico, vous devez le configurer `installation.calicoNetwork.bgp: Enabled` dans votre configuration Helm. Pour plus d'informations sur le support BGP de Calico, consultez Configurer le peering BGP dans la documentation de Calico.
- La gestion des adresses IP (IPAM) par défaut de Calico s'appelle Calico IPAM, où le `calico-ipam` plugin alloue des adresses IP à chaque nœud en fonction du pod configuré par l'utilisateur. CIDRs Les pods CIDRs sont configurés avec la valeur `installation.calicoNetwork.ipPools.cidr` Helm, qui doit correspondre au réseau de pods distants CIDRs que vous avez configuré pour votre cluster Amazon EKS. Calico alloue des segments de à chaque nœud`ipPools.cidr`. La taille des segments par nœud est configurée avec la valeur `ipPools.blockSize` Helm. Pour plus d'informations sur l'IPAM avec Calico, consultez Commencer à gérer les adresses IP dans la documentation de Calico.

Installez Calico sur des nœuds hybrides

1. Assurez-vous d'avoir installé la CLI helm sur votre environnement de ligne de commande. Consultez la documentation Helm pour les instructions d'installation.
2. Installez le dépôt Cilium Helm.

```
helm repo add projectcalico https://docs.tigera.io/calico/charts
```

3. Créez un fichier YAML appelé`calico-values.yaml`. L'exemple suivant configure tous les composants Calico pour qu'ils s'exécutent uniquement sur des nœuds hybrides en définissant l'affinité pour l'`eks.amazonaws.com/compute-type: hybrid`étiquette.
   - `POD_CIDR`Remplacez-les par les plages CIDR de vos pods. Si vous avez configuré votre cluster Amazon EKS avec des réseaux de pods distants, `POD_CIDR` ceux que vous spécifiez pour Calico doivent être identiques aux réseaux de pods distants. Par exemple, `10.100.0.0/24`. Le CIDR de votre pod local ne doit pas chevaucher le CIDR de votre nœud local lorsque vous exécutez le CNI en mode overlay/tunnel.
   - Remplacez `CIDR_SIZE` par la taille du segment CIDR que vous souhaitez allouer à chaque nœud. Par exemple, 25 pour une taille de segment /25 de 128 adresses de pod par nœud. Pour plus d'informations sur le CIDR `blockSize` et la modification du`blockSize`, voir Modifier la taille de bloc du pool IP dans la documentation de Calico.
   - Dans l'exemple ci-dessous, `natOutgoing` est activé et bgp est désactivé. Modifiez ces valeurs en fonction de votre configuration cible.

```
installation:
  enabled: true
  cni:
    type: Calico
    ipam:
      type: Calico
  calicoNetwork:
    bgp: Disabled
    ipPools:
    - cidr: POD_CIDR
      blockSize: CIDR_SIZE
      encapsulation: VXLAN
      natOutgoing: Enabled
      nodeSelector: eks.amazonaws.com/compute-type == "hybrid"
  controlPlaneReplicas: 1
  controlPlaneNodeSelector:
    eks.amazonaws.com/compute-type: hybrid
  calicoNodeDaemonSet:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid
  csiNodeDriverDaemonSet:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid
  typhaDeployment:
    spec:
      template:
        spec:
          nodeSelector:
            eks.amazonaws.com/compute-type: hybrid
```

4. Installez Calico sur votre cluster.

- `CALICO_VERSION`Remplacez-le par une version Calico (par exemple`v3.29.0`), consultez les versions de Calico pour trouver le dernier correctif pour votre version mineure de Calico. Nous vous recommandons d'exécuter la dernière version du correctif pour la version mineure de Calico.
- Si vous utilisez un `kubeconfig` fichier spécifique, utilisez le `--kubeconfig` drapeau.

```
helm install calico projectcalico/tigera-operator \
    --version CALICO_VERSION \
    --namespace kube-system \
    -f calico-values.yaml
```

5. Vous pouvez vérifier que votre installation de Calico s'est bien déroulée à l'aide des commandes suivantes. Vous devriez voir le `tigera-operator` déploiement, l'`calico-node`agent exécuté sur chacun de vos nœuds hybrides `calico-apiserver`, `csi-node-driver`, le et `calico-kube-controllers` déployé. De plus, vos nœuds hybrides devraient désormais avoir un statut`Ready`. Si vous en utilisez`natOutgoing: Disabled`, tous les composants Calico ne pourront pas démarrer correctement tant que vous n'aurez pas annoncé les adresses de vos pods sur votre réseau local. Pour plus d'informations sur la configuration de BGP pour Calico, passez à l'étape suivante.

```
kubectl get pods -A
```

```
NAMESPACE          NAME                                       READY   STATUS
 RESTARTS   AGE
calico-apiserver   calico-apiserver-6c77bb6d46-2n8mq          1/1     Running   0
      69s
calico-system      calico-kube-controllers-7c5f8556b5-7h267   1/1     Running   0
      68s
calico-system      calico-node-s5nnk                          1/1     Running   0
      68s
calico-system      calico-typha-6487cc9d8c-wc5jm              1/1     Running   0
      69s
calico-system      csi-node-driver-cv42d                      2/2     Running   0
      68s
kube-system        coredns-7bb495d866-2lc9v                   1/1     Running   0
      6m27s
kube-system        coredns-7bb495d866-2t8ln                   1/1     Running   0
      157m
kube-system        kube-proxy-lxzxh                           1/1     Running   0
      18m
```

```
kube-system        tigera-operator-f8bc97d4c-28b4d            1/1     Running   0
      90s
```

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES    AGE    VERSION
mi-0c6ec2f6f79176565   Ready    <none>   5h13m  v1.31.0-eks-a737599
```

6. Si vous avez installé Calico sans BGP, ignorez cette étape. Pour configurer BGP, créez un fichier appelé `calico-bgp.yaml` avec une `BGPPeer` configuration et un`BGPConfiguration`. Il est important de distinguer `BGPPeer` et`BGPConfiguration`. `BGPPeer`Il s'agit du routeur compatible BGP ou de la ressource distante avec laquelle les nœuds d'un cluster Calico seront homologues. La `BGPPeer` configuration `asNumber` in est similaire au réglage Cilium. `peerASN` Le `BGPConfiguration` est appliqué à chaque nœud Calico et le `asNumber` for `BGPConfiguration` est équivalent au paramètre Cilium. `localASN` Remplacez `ONPREM_ROUTER_IP`ONPREM_ROUTER_ASN, et `LOCAL_ASN` dans l'exemple ci-dessous par les valeurs de votre environnement sur site, que vous devrez peut-être obtenir auprès de votre administrateur réseau. Le `keepOriginalNextHop: true` paramètre est utilisé pour garantir que chaque nœud annonce uniquement le CIDR du réseau de pods dont il est propriétaire.

```
apiVersion: projectcalico.org/v3
kind: BGPPeer
metadata:
  name: calico-hybrid-nodes
spec:
  peerIP: ONPREM_ROUTER_IP
  asNumber: ONPREM_ROUTER_ASN
  keepOriginalNextHop: true
---
apiVersion: projectcalico.org/v3
kind: BGPConfiguration
metadata:
  name: default
spec:
  nodeToNodeMeshEnabled: false
  asNumber: LOCAL_ASN
```

7. Appliquez le fichier à votre cluster.

```
kubectl apply -f calico-bgp.yaml
```

8. Vérifiez que les pods Calico sont en cours d'exécution à l'aide de la commande suivante.

```
kubectl get pods -n calico-system -w
```

```
NAMESPACE          NAME                                       READY   STATUS
 RESTARTS       AGE
calico-apiserver   calico-apiserver-598bf99b6c-2vltk          1/1     Running   0
          3h24m
calico-system      calico-kube-controllers-75f84bbfd6-zwmnx   1/1     Running   31
 (59m ago)   3h20m
calico-system      calico-node-9b2pg                          1/1     Running   0
          5h17m
calico-system      calico-typha-7d55c76584-kxtnq              1/1     Running   0
          5h18m
calico-system      csi-node-driver-dmnmm                      2/2     Running   0
          5h18m
kube-system        coredns-7bb495d866-dtn4z                   1/1     Running   0
          6h23m
kube-system        coredns-7bb495d866-mk7j4                   1/1     Running   0
          6h19m
kube-system        kube-proxy-vms28                           1/1     Running   0
          6h12m
kube-system        tigera-operator-55f9d9d565-jj9bg           1/1     Running   0
          73m
```

Si vous avez rencontré des problèmes au cours de ces étapes, consultez les instructions de dépannage dans la documentation de Calico.

Mettez à niveau Calico sur des nœuds hybrides

Avant de mettre à niveau votre déploiement Calico, consultez attentivement la documentation de mise à niveau de Calico et les notes de publication pour comprendre les modifications apportées à la version cible de Calico. Les étapes de mise à niveau varient selon que vous utilisez Helm, l'opérateur Calico et le type de banque de données. Les étapes ci-dessous supposent l'utilisation de Helm.

1. Téléchargez le manifeste de l'opérateur pour la version de Calico vers laquelle vous effectuez la mise à niveau. `CALICO_VERSION`Remplacez-le par la version vers laquelle vous effectuez la mise à niveau, par exemple`v3.29.0`. Assurez-vous d'ajouter le au fichier v major.minor.patch.

```
kubectl apply --server-side --force-conflicts \
    -f https://raw.githubusercontent.com/projectcalico/calico/CALICO_VERSION/
manifests/operator-crds.yaml
```

2. Exécutez `helm upgrade` pour mettre à niveau votre déploiement Calico. `CALICO_VERSION`Remplacez-le par la version vers laquelle vous effectuez la mise à niveau, par exemple`v3.29.0`. Créez le `calico-values.yaml` fichier à partir des valeurs de configuration que vous avez utilisées pour installer Calico.

```
helm upgrade calico projectcalico/tigera-operator \
    --version CALICO_VERSION \
    --namespace kube-system \
    -f calico-values.yaml
```

Supprimer Calico des nœuds hybrides

1. Exécutez la commande suivante pour désinstaller les composants Calico de votre cluster. Notez que la désinstallation du CNI peut avoir un impact sur l'état des nœuds et des pods et ne doit pas être effectuée sur des clusters de production. Si vous avez installé Calico dans un espace de noms autre que le `kube-system` changement d'espace de noms dans la commande ci-dessous.

```
helm uninstall calico --namespace kube-system
```

   Notez que les interfaces et les routes configurées par Calico ne sont pas supprimées par défaut lorsque vous supprimez le CNI du cluster.

2. Pour nettoyer les fichiers de configuration et les ressources sur disque, supprimez les fichiers Calico des répertoires `/opt/cni` et`/etc/cni`.

3. Pour supprimer le Calico CRDs de votre cluster, exécutez les commandes suivantes.

```
kubectl get crds -oname | grep "calico" | xargs kubectl delete
```

```
kubectl get crds -oname | grep "tigera" | xargs kubectl delete
```

Configuration des modules complémentaires pour les nœuds hybrides

Cette page décrit les considérations relatives à l'exécution de AWS modules complémentaires et de modules complémentaires communautaires sur les nœuds hybrides Amazon EKS. Pour en savoir plus sur les modules complémentaires Amazon EKS et les processus de création, de mise à niveau et de suppression des modules complémentaires de votre cluster, consultezthe section called "Modules complémentaires Amazon EKS". Sauf indication contraire sur cette page, les processus de création, de mise à niveau et de suppression des modules complémentaires Amazon EKS sont les mêmes pour les clusters Amazon EKS dotés de nœuds hybrides que pour les clusters Amazon EKS dotés de nœuds exécutés dans AWS le cloud. Seuls les modules complémentaires inclus sur cette page ont été validés pour leur compatibilité avec Amazon EKS Hybrid Nodes.

Les AWS modules complémentaires suivants sont compatibles avec les nœuds hybrides Amazon EKS.

AWS module complémentaire	Versions complémentaires compatibles
kube-proxy	v1.25.14-eksbuild.2 et versions ultérieures
CoreDNS	v1.9.3-eksbuild.7 et versions ultérieures
AWS Distro pour OpenTelemetry (ADOT)	v0.102.1-eksbuild.2 et versions ultérieures
CloudWatch Agent d'observabilité	v2.2.1-eksbuild.1 et versions ultérieures
Agent d'identité du pod EKS	• v1.3.3-eksbuild.1 et versions ultérieures, sauf pour Bottlerocket • v1.3.7-eksbuild.2 et versions ultérieures pour Bottlerocket
Agent de surveillance des nœuds	v1.2.0-eksbuild.1 et versions ultérieures
Contrôleur d'instantané CSI	v8.1.0-eksbuild.1 et versions ultérieures
AWS Connecteur CA privé pour Kubernetes	v1.6.0-eksbuild.1 et versions ultérieures

Les modules complémentaires communautaires suivants sont compatibles avec les nœuds hybrides Amazon EKS. Pour en savoir plus sur les modules complémentaires communautaires, consultezthe section called "Modules complémentaires communautaires".

Module complémentaire communautaire	Versions complémentaires compatibles
Serveur de métriques Kubernetes	v0.7.2-eksbuild.1 et versions ultérieures
gestionnaire de certificats	v1.17.2-eksbuild.1 et versions ultérieures
Prometheus Node Exporter	v1.9.1-eksbuild.2 et versions ultérieures
kube-state-metrics	v2.15.0-eksbuild.4 et versions ultérieures

Outre les modules complémentaires Amazon EKS présentés dans les tableaux ci-dessus, Amazon Managed Service pour Prometheus Collector et le Load AWS Balancer Controller pour l'entrée d'applications (HTTP) et l'équilibrage de charge (TCP/UDP) sont compatibles avec les nœuds hybrides.

Il existe des AWS modules complémentaires et des modules complémentaires communautaires qui ne sont pas compatibles avec les nœuds hybrides Amazon EKS. Les dernières versions de ces modules complémentaires disposent d'une règle anti-affinité pour l'`eks.amazonaws.com/compute-type: hybrid`étiquette par défaut appliquée aux nœuds hybrides. Cela les empêche de s'exécuter sur des nœuds hybrides lorsqu'ils sont déployés dans vos clusters. Si vous avez des clusters comportant à la fois des nœuds hybrides et des nœuds exécutés dans AWS le cloud, vous pouvez déployer ces modules complémentaires dans votre cluster sur des nœuds exécutés dans AWS le cloud. L'Amazon VPC CNI n'est pas compatible avec les nœuds hybrides, et Cilium et Calico sont pris en charge en tant qu'interfaces réseau de conteneurs () CNIs pour les nœuds hybrides Amazon EKS. Pour plus d'informations, consultez the section called "Configurer CNI".

AWS modules complémentaires

Les sections suivantes décrivent les différences entre l'exécution de AWS modules complémentaires compatibles sur des nœuds hybrides par rapport aux autres types de calcul Amazon EKS.

kube-proxy et CoreDNS

EKS installe kube-proxy et CoreDNS en tant que modules complémentaires autogérés par défaut lorsque vous créez un cluster EKS avec l'API AWS et, notamment, à partir de la CLI. AWS SDKs AWS Vous pouvez remplacer ces modules complémentaires par des modules complémentaires Amazon EKS après la création du cluster. Consultez la documentation EKS pour plus de détails sur the section called "kube-proxy" etthe section called "CoreDNS". Si vous utilisez un cluster

en mode mixte avec à la fois des nœuds hybrides et des nœuds dans le AWS Cloud, nous vous recommandons de disposer d'au moins une réplique CoreDNS sur les nœuds hybrides et d'au moins une réplique CoreDNS sur vos nœuds dans le Cloud. AWS Consultez the section called "Configuration des répliques CoreDNS" les étapes de configuration.

CloudWatch Agent d'observabilité

L'opérateur de l'agent CloudWatch Observability utilise des webhooks. Si vous exécutez l'opérateur sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local et vous devez configurer votre cluster EKS avec votre réseau de pods distants. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Les métriques au niveau des nœuds ne sont pas disponibles pour les nœuds hybrides car CloudWatch Container Insights dépend de la disponibilité du service IMDS (Instance Metadata Service) pour les métriques au niveau des nœuds. Les métriques au niveau du cluster, de la charge de travail, du pod et du conteneur sont disponibles pour les nœuds hybrides.

Après avoir installé le module complémentaire en suivant les étapes décrites dans Installer l' CloudWatch agent avec Amazon CloudWatch Observability, le manifeste du module complémentaire doit être mis à jour pour que l'agent puisse s'exécuter correctement sur des nœuds hybrides. Modifiez la `amazoncloudwatchagents` ressource du cluster pour ajouter la variable d'`RUN_WITH_IRSA`environnement comme indiqué ci-dessous.

```
kubectl edit amazoncloudwatchagents -n amazon-cloudwatch cloudwatch-agent
```

```
apiVersion: v1
items:
- apiVersion: cloudwatch.aws.amazon.com/v1alpha1
  kind: AmazonCloudWatchAgent
  metadata:
    ...
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
    ...
  spec:
    ...
    env:
    - name: RUN_WITH_IRSA # <-- Add this
      value: "True" # <-- Add this
    - name: K8S_NODE_NAME
```

```
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
          ...
```

Collecteur géré par Amazon Managed Prometheus pour nœuds hybrides

Un collecteur géré par Amazon Managed Service for Prometheus (AMP) consiste en un scraper qui découvre et collecte des métriques à partir des ressources d'un cluster Amazon EKS. AMP gère le scraper pour vous, vous évitant ainsi de devoir gérer vous-même les instances, les agents ou les scrapers.

Vous pouvez utiliser les collecteurs gérés par AMP sans configuration supplémentaire spécifique aux nœuds hybrides. Cependant, les points de terminaison métriques de vos applications sur les nœuds hybrides doivent être accessibles depuis le VPC, y compris les itinéraires entre le VPC et le CIDRs réseau de pods distants et les ports ouverts dans votre pare-feu sur site. En outre, votre cluster doit disposer d'un accès de point de terminaison privé.

Suivez les étapes décrites dans la section Utilisation d'un collecteur AWS géré dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

AWS Distro pour OpenTelemetry (ADOT)

Vous pouvez utiliser le module complémentaire AWS Distro for OpenTelemetry (ADOT) pour collecter des métriques, des journaux et des données de suivi à partir de vos applications exécutées sur des nœuds hybrides. ADOT utilise des webhooks d'admission pour muter et valider les demandes de ressources personnalisées du collecteur. Si vous exécutez l'opérateur ADOT sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local et vous devez configurer votre cluster EKS avec votre réseau de pods distants. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Suivez les étapes décrites dans Getting Started with AWS Distro pour OpenTelemetry utiliser les modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

AWS Contrôleur Load Balancer

Vous pouvez utiliser le AWS Load Balancer Controller et l'Application Load Balancer (ALB) ou le Network Load Balancer (NLB) avec l'adresse IP de type cible pour les charges de travail sur des

nœuds hybrides connectés à Direct Connect ou à un VPN. AWS AWS Site-to-Site La ou les cibles IP utilisées avec l'ALB ou le NLB doivent être routables depuis. AWSLe contrôleur AWS Load Balancer utilise également des webhooks. Si vous exécutez l'opérateur AWS Load Balancer Controller sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local et vous devez configurer votre cluster EKS avec votre réseau de pods distants. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Pour installer le AWS Load Balancer Controller, suivez les étapes indiquées dans ou. the section called "Installation avec Helm" the section called "Installation avec des manifestes"

Pour l'entrée avec ALB, vous devez spécifier les annotations ci-dessous. Pour obtenir des instructions, consultez the section called "Répartition de la charge des applications".

```
alb.ingress.kubernetes.io/target-type: ip
```

Pour l'équilibrage de charge avec NLB, vous devez spécifier les annotations ci-dessous. Pour obtenir des instructions, consultez the section called "Répartition de charge réseau".

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Agent d'identité du pod EKS

> **Note**
>
> Pour déployer avec succès le module complémentaire EKS Pod Identity Agent sur des nœuds hybrides exécutant Bottlerocket, assurez-vous que votre version de Bottlerocket est au moins la version v1.39.0. Le Pod Identity Agent n'est pas pris en charge sur les versions antérieures de Bottlerocket dans les environnements de nœuds hybrides.

L'agent d'identité Amazon EKS Pod d'origine DaemonSet s'appuie sur la disponibilité de l' EC2 IMDS sur le nœud pour obtenir les informations d' AWS identification requises. Comme l'IMDS n'est pas disponible sur les nœuds hybrides, à partir de la version 1.3.3-eksbuild.1, le module complémentaire Pod Identity Agent déploie éventuellement un module qui monte les informations d'identification requises. DaemonSet Les nœuds hybrides exécutant Bottlerocket nécessitent une méthode différente pour monter les informations d'identification, et à partir de la version 1.3.7-eksbuild.2, le module complémentaire Pod Identity Agent déploie éventuellement une méthode qui cible spécifiquement les

nœuds hybrides Bottlerocket. DaemonSet Les sections suivantes décrivent le processus d'activation de l'option DaemonSets.

Ubuntu/RHEL/AL2023

1. Pour utiliser l'agent Pod Identity sur Ubuntu/RHEL/Al 2023 nœuds hybrides, `enableCredentialsFile: true` configurez-le dans la section hybride de la `nodeadm` configuration comme indiqué ci-dessous :

   ```
   apiVersion: node.eks.aws/v1alpha1
   kind: NodeConfig
   spec:
       hybrid:
           enableCredentialsFile: true # <-- Add this
   ```

   Cela permettra `nodeadm` de créer un fichier d'informations d'identification à configurer sur le nœud ci-dessous`/eks-hybrid/.aws/credentials`, qui sera utilisé par les `eks-pod-identity-agent` pods. Ce fichier d'informations d'identification contiendra des AWS informations d'identification temporaires qui seront actualisées périodiquement.

2. Après avoir mis à jour la `nodeadm` configuration sur chaque nœud, exécutez la `nodeadm init` commande suivante `nodeConfig.yaml` pour joindre vos nœuds hybrides à votre cluster Amazon EKS. Si vos nœuds ont déjà rejoint le cluster, réexécutez toujours la `init` commande.

   ```
   nodeadm init -c file://nodeConfig.yaml
   ```

3. Installation `eks-pod-identity-agent` avec prise en charge des nœuds hybrides activée, à l'aide de la AWS CLI ou AWS Management Console.

   a. AWS CLI : depuis la machine que vous utilisez pour administrer le cluster, exécutez la commande suivante pour effectuer l'installation en `eks-pod-identity-agent` activant la prise en charge des nœuds hybrides. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
       --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
   ```

   b. AWS Management Console: Si vous installez le module complémentaire Pod Identity Agent via la AWS console, ajoutez ce qui suit à la configuration facultative pour déployer DaemonSet celui qui cible les nœuds hybrides.

```
{"daemonsets":{"hybrid":{"create": true}}}
```

Bottlerocket

1. Pour utiliser l'agent Pod Identity sur les nœuds hybrides Bottlerocket, ajoutez l'`--enable-credentials-file=true`indicateur à la commande utilisée pour les données utilisateur du conteneur Bottlerocket bootstrap, comme décrit dans. [the section called "Connectez des nœuds hybrides avec Bottlerocket"]

   a. Si vous utilisez le fournisseur d'informations d'identification SSM, votre commande doit ressembler à ceci :

   ```
   eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
   code=<activation-code> --region=<region> --enable-credentials-file=true
   ```

   b. Si vous utilisez le fournisseur d'informations d'identification IAM Roles Anywhere, votre commande doit se présenter comme suit :

   ```
   eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
   credentials-file=true
   ```

   Cela configurera le script bootstrap pour créer un fichier d'informations d'identification sur le nœud situé en dessous`/var/eks-hybrid/.aws/credentials`, qui sera utilisé par les `eks-pod-identity-agent` pods. Ce fichier d'informations d'identification contiendra des AWS informations d'identification temporaires qui seront actualisées périodiquement.

2. Installation `eks-pod-identity-agent` avec prise en charge des nœuds hybrides Bottlerocket activée, à l'aide de la CLI AWS ou. AWS Management Console

   a. AWS CLI : depuis la machine que vous utilisez pour administrer le cluster, exécutez la commande suivante pour effectuer l'installation en activant la prise en charge `eks-pod-identity-agent` des nœuds hybrides Bottlerocket. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
       --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
    true}}}'
   ```

b. AWS Management Console: Si vous installez le module complémentaire Pod Identity Agent via la AWS console, ajoutez ce qui suit à la configuration facultative pour déployer le module DaemonSet qui cible les nœuds hybrides Bottlerocket.

```
{"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
```

Contrôleur d'instantané CSI

À partir de la version`v8.1.0-eksbuild.2`, le module complémentaire CSI Snapshot Controller applique une règle d'anti-affinité souple pour les nœuds hybrides, préférant que le contrôleur `deployment` s'exécute EC2 dans la même AWS région que le plan de contrôle Amazon EKS. La colocation `deployment` dans la même AWS région que le plan de contrôle Amazon EKS améliore la latence.

Modules complémentaires communautaires

Les sections suivantes décrivent les différences entre l'exécution de modules complémentaires communautaires compatibles sur des nœuds hybrides par rapport aux autres types de calcul Amazon EKS.

Serveur de métriques Kubernetes

Le plan de contrôle doit atteindre l'adresse IP du pod du serveur Metrics (ou l'adresse IP du nœud si HostNetwork est activé). Par conséquent, à moins que vous n'exécutiez Metrics Server en mode HostNetwork, vous devez configurer un réseau de pods distants lors de la création de votre cluster Amazon EKS, et vous devez rendre les adresses IP de vos pods routables. La mise en œuvre du protocole BGP (Border Gateway Protocol) avec le CNI est un moyen courant de rendre les adresses IP de votre pod routables.

gestionnaire de certificats

`cert-manager`utilise des webhooks. Si vous exécutez `cert-manager` sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local et vous devez configurer votre cluster EKS avec votre réseau de pods distants. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Configurer les webhooks pour les nœuds hybrides

Cette page détaille les considérations relatives à l'exécution de webhooks avec des nœuds hybrides. Les webhooks sont utilisés dans les applications Kubernetes et les projets open source, tels

que le AWS Load Balancer Controller et l' CloudWatch Observability Agent, pour effectuer des fonctionnalités de mutation et de validation lors de l'exécution.

Réseaux de pods routables

Si vous parvenez à rendre votre pod CIDR local routable sur votre réseau local, vous pouvez exécuter des webhooks sur des nœuds hybrides. Vous pouvez utiliser plusieurs techniques pour rendre votre pod CIDR local routable sur votre réseau local, notamment le Border Gateway Protocol (BGP), les itinéraires statiques ou d'autres solutions de routage personnalisées. Le BGP est la solution recommandée car elle est plus évolutive et plus facile à gérer que les solutions alternatives qui nécessitent une configuration de routage personnalisée ou manuelle. AWS prend en charge les fonctionnalités BGP de Cilium et Calico pour les modules publicitaires CIDRs, voir the section called "Configurer CNI" et the section called "Pod à distance routable CIDRs" pour plus d'informations.

Réseaux de pods non routables

Si vous ne pouvez pas rendre votre pod CIDR sur site routable sur votre réseau local et que vous devez exécuter des webhooks, nous vous recommandons d'exécuter tous les webhooks sur des nœuds cloud du même cluster EKS que vos nœuds hybrides.

Considérations relatives aux clusters en mode mixte

Les clusters en mode mixte sont définis comme des clusters EKS comportant à la fois des nœuds hybrides et des nœuds exécutés dans AWS le cloud. Lorsque vous exécutez un cluster en mode mixte, tenez compte des recommandations suivantes :

- Exécutez le VPC CNI sur des nœuds dans le AWS cloud et Cilium ou Calico sur des nœuds hybrides. Cilium et Calico ne sont pas pris en charge AWS lorsqu'ils sont exécutés sur des nœuds dans le Cloud. AWS
- Configurez les webhooks pour qu'ils s'exécutent sur des nœuds dans le AWS cloud. Découvrez the section called "Configurer les webhooks pour les modules complémentaires" comment configurer les webhooks AWS et les modules complémentaires communautaires.
- Si vos applications nécessitent des pods exécutés sur des nœuds dans le AWS cloud pour communiquer directement avec des pods exécutés sur des nœuds hybrides (« communication est-ouest »), et que vous utilisez le VPC CNI sur des nœuds dans le AWS cloud, et Cilium ou Calico sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local.
- Exécutez au moins une réplique de CoreDNS sur des nœuds AWS dans le Cloud et au moins une réplique de CoreDNS sur des nœuds hybrides.

- Configurez la distribution du trafic de service pour que le trafic de service reste local dans la zone d'où il provient. Pour plus d'informations sur la distribution du trafic de service, consultezthe section called "Configuration de la distribution du trafic de service".
- Si vous utilisez des équilibreurs de charge d' AWS application (ALB) ou des équilibreurs de charge réseau (NLB) pour le trafic de charge de travail exécuté sur des nœuds hybrides, les cibles IP utilisées avec l'ALB ou le NLB doivent être routables depuis. AWS
- Le module complémentaire Metrics Server nécessite une connectivité entre le plan de contrôle EKS et l'adresse IP du pod Metrics Server. Si vous exécutez le module complémentaire Metrics Server sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local.
- Pour collecter des métriques pour les nœuds hybrides à l'aide de collecteurs gérés par Amazon Managed Service for Prometheus (AMP), le CIDR de votre pod local doit être routable sur votre réseau local. Vous pouvez également utiliser le collecteur géré AMP pour les métriques et les ressources du plan de contrôle EKS exécutées dans le AWS cloud, et le module complémentaire AWS Distro for OpenTelemetry (ADOT) pour collecter des métriques pour les nœuds hybrides.

Configuration de clusters en mode mixte

Pour visualiser les webhooks mutants et validants exécutés sur votre cluster, vous pouvez consulter le type de ressource Extensions dans le panneau Ressources de la console EKS correspondant à votre cluster, ou vous pouvez utiliser les commandes suivantes. EKS indique également les métriques du webhook dans le tableau de bord d'observabilité du cluster, voir the section called "Tableau de bord d'observabilité" pour plus d'informations.

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

Configuration de la distribution du trafic de service

Lorsque vous exécutez des clusters en mode mixte, nous vous recommandons d'utiliser la distribution du trafic de service pour que le trafic de service reste local dans la zone d'où il provient. La distribution du trafic de service (disponible pour les versions 1.31 et ultérieures de Kubernetes dans EKS) est la solution recommandée par rapport au routage basé sur la topologie, car elle est plus prévisible. Avec la distribution du trafic de service, les terminaux sains de la zone recevront tout le trafic de cette zone. Avec le routage adapté à la topologie, chaque service doit remplir plusieurs

conditions dans cette zone pour appliquer le routage personnalisé, sinon il achemine le trafic de manière uniforme vers tous les points de terminaison.

Si vous utilisez Cilium comme CNI, vous devez exécuter le CNI avec le paramètre `enable-service-topology` défini sur pour activer la distribution du trafic `true` de service. Vous pouvez transmettre cette configuration à l'aide de l'indicateur d'installation Helm `--set loadBalancer.serviceTopology=true` ou vous pouvez mettre à jour une installation existante à l'aide de la commande Cilium CLI. `cilium config set enable-service-topology true` L'agent Cilium exécuté sur chaque nœud doit être redémarré après la mise à jour de la configuration d'une installation existante.

Un exemple de configuration de la distribution du trafic de service pour le service CoreDNS est présenté dans la section suivante, et nous vous recommandons d'activer la même méthode pour tous les services de votre cluster afin d'éviter le trafic interenvironnemental involontaire.

Configuration des répliques CoreDNS

Lorsque vous exécutez des clusters en mode mixte, nous vous recommandons de disposer d'au moins une réplique CoreDNS sur les nœuds hybrides et d'au moins une réplique CoreDNS sur les nœuds du Cloud. AWS

1. Ajoutez une étiquette de zone topologique pour chacun de vos nœuds hybrides, par exemple`topology.kubernetes.io/zone: onprem`. Vous pouvez également définir l'étiquette lors de la `nodeadm init` phase en spécifiant l'étiquette dans votre `nodeadm` configuration, voirthe section called “Config de nœud pour personnaliser kubelet (facultatif)”. Notez que les nœuds exécutés dans AWS le cloud reçoivent automatiquement une étiquette de zone topologique qui correspond à la zone de disponibilité (AZ) du nœud.

   ```
   kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
   ```

2. Ajoutez `podAntiAffinity` au déploiement CoreDNS avec la clé de zone topologique. Vous pouvez également configurer le déploiement de CoreDNS lors de l'installation à l'aide des modules complémentaires EKS.

   ```
   kubectl edit deployment coredns -n kube-system
   ```

   ```
   spec:
     template:
   ```

```
    spec:
      affinity:
       ...
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: kubernetes.io/hostname
            weight: 100
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: topology.kubernetes.io/zone
            weight: 50
      ...
```

3. Ajoutez le paramètre `trafficDistribution: PreferClose` à la configuration du `kube-dns` service pour activer la distribution du trafic du service.

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
}'
```

4. Vous pouvez confirmer que la distribution du trafic de service est activée en consultant les tranches de point de terminaison du `kube-dns` service. Vos tranches de point de terminaison doivent afficher `hints` les étiquettes de votre zone topologique, ce qui confirme que la distribution du trafic de service est activée. Si vous ne voyez pas l'adresse de chaque point `hints` de terminaison, cela signifie que la distribution du trafic de service n'est pas activée.

```
kubectl get endpointslice -A | grep "kube-dns"
```

```
kubectl get endpointslice [.replaceable]`kube-dns-<id>`  -n kube-system -o yaml
```

```
addressType: IPv4
apiVersion: discovery.k8s.io/v1
endpoints:
- addresses:
  - <your-hybrid-node-pod-ip>
  hints:
    forZones:
    - name: onprem
  nodeName: <your-hybrid-node-name>
  zone: onprem
- addresses:
  - <your-cloud-node-pod-ip>
  hints:
    forZones:
    - name: us-west-2a
  nodeName: <your-cloud-node-name>
  zone: us-west-2a
```

Configurer les webhooks pour les modules complémentaires

Les modules complémentaires suivants utilisent des webhooks et sont compatibles avec les nœuds hybrides.

- AWS Contrôleur Load Balancer
- CloudWatch Agent d'observabilité
- AWS Distribution pour OpenTelemetry (ADOT)
- `cert-manager`

Consultez les sections suivantes pour configurer les webhooks utilisés par ces modules complémentaires pour qu'ils s'exécutent sur des nœuds dans le AWS Cloud.

AWS Contrôleur Load Balancer

Pour utiliser le AWS Load Balancer Controller dans une configuration de cluster en mode mixte, vous devez exécuter le contrôleur sur des nœuds du Cloud. AWS Pour ce faire, ajoutez ce qui suit à votre

configuration des valeurs Helm ou spécifiez les valeurs à l'aide de la configuration complémentaire EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

CloudWatch Agent d'observabilité

Le module complémentaire CloudWatch Observability Agent possède un opérateur Kubernetes qui utilise des webhooks. Pour exécuter l'opérateur sur des nœuds du AWS cloud dans une configuration de cluster en mode mixte, modifiez la configuration de l'opérateur CloudWatch Observability Agent. Vous ne pouvez pas configurer l'affinité des opérateurs lors de l'installation avec les modules complémentaires Helm et EKS (voir le problème #2431 de containers-roadmap).

```
kubectl edit -n amazon-cloudwatch deployment amazon-cloudwatch-observability-
controller-manager
```

```
spec:
  ...
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                - hybrid
```

AWS Distribution pour OpenTelemetry (ADOT)

Le module complémentaire AWS Distro for OpenTelemetry (ADOT) possède un opérateur Kubernetes qui utilise des webhooks. Pour exécuter l'opérateur sur des nœuds dans le AWS cloud dans une configuration de cluster en mode mixte, ajoutez ce qui suit à votre configuration des valeurs Helm ou spécifiez les valeurs à l'aide de la configuration complémentaire EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Si le CIDR de votre pod n'est pas routable sur votre réseau local, le collecteur ADOT doit s'exécuter sur des nœuds hybrides pour extraire les métriques de vos nœuds hybrides et des charges de travail qui s'y exécutent. Pour ce faire, modifiez la définition de ressource personnalisée (CRD).

```
kubectl -n opentelemetry-operator-system edit opentelemetrycollectors.opentelemetry.io
 adot-col-prom-metrics
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

Vous pouvez configurer le collecteur ADOT pour extraire uniquement les métriques des nœuds hybrides et des ressources exécutées sur les nœuds hybrides en ajoutant les éléments suivants `relabel_configs` à chacun `scrape_configs` dans la configuration CRD du collecteur ADOT.

```
relabel_configs:
```

```
  - action: keep
    regex: hybrid
    source_labels:
    - __meta_kubernetes_node_label_eks_amazonaws_com_compute_type
```

Le module complémentaire ADOT doit être installé `cert-manager` pour les certificats TLS utilisés par le webhook de l'opérateur ADOT. `cert-manager`exécute également des webhooks et vous pouvez le configurer pour qu'il s'exécute sur des nœuds dans le AWS cloud avec la configuration des valeurs Helm suivante.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
```

```
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

`cert-manager`

Le `cert-manager` module complémentaire exécute des webhooks et vous pouvez le configurer pour qu'il s'exécute sur des nœuds dans le AWS cloud avec la configuration des valeurs Helm suivante.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
```

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

Configuration du proxy pour les nœuds hybrides

Si vous utilisez un serveur proxy dans votre environnement sur site pour le trafic sortant de votre centre de données ou de votre environnement périphérique, vous devez configurer séparément vos nœuds et votre cluster pour utiliser votre serveur proxy.

Cluster

Sur votre cluster, vous devez configurer `kube-proxy` pour utiliser votre serveur proxy. Vous devez le configurer `kube-proxy` après avoir créé votre cluster Amazon EKS.

Nœuds

Sur vos nœuds, vous devez configurer le système d'exploitation`containerd`,`kubelet`, et l'agent Amazon SSM pour utiliser votre serveur proxy. Vous pouvez apporter ces modifications au cours du processus de création des images de votre système d'exploitation ou avant de `nodeadm init` les exécuter sur chaque nœud hybride.

Configuration au niveau du nœud

Vous devez appliquer les configurations suivantes soit dans les images de votre système d'exploitation, soit avant de `nodeadm init` les exécuter sur chaque nœud hybride.

`containerd`configuration du proxy

`containerd`est l'environnement d'exécution de gestion de conteneurs par défaut pour Kubernetes. Si vous utilisez un proxy pour accéder à Internet, vous devez le configurer de manière à ce qu'il `containerd` puisse extraire les images de conteneur requises par Kubernetes et Amazon EKS.

Créez un fichier sur chaque nœud hybride appelé `http-proxy.conf` dans le `/etc/systemd/system/containerd.service.d` répertoire avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs de votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

`containerd`configuration à partir des données utilisateur

Le `containerd.service.d` répertoire devra être créé pour ce fichier. Vous devrez recharger systemd pour récupérer le fichier de configuration sans redémarrer. En AL2 023, le service sera probablement déjà en cours d'exécution lorsque votre script s'exécutera. Vous devrez donc également le redémarrer.

```
mkdir -p /etc/systemd/system/containerd.service.d
echo '[Service]' > /etc/systemd/system/containerd.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/containerd.service.d/
http-proxy.conf
systemctl daemon-reload
systemctl restart containerd
```

`kubelet`configuration du proxy

`kubelet`est l'agent de nœud Kubernetes qui s'exécute sur chaque nœud Kubernetes et est chargé de gérer le nœud et les pods qui y sont exécutés. Si vous utilisez un proxy dans votre environnement sur site, vous devez le configurer `kubelet` afin qu'il puisse communiquer avec les points de terminaison publics ou privés de votre cluster Amazon EKS.

Créez un fichier sur chaque nœud hybride appelé `http-proxy.conf` dans le `/etc/systemd/system/kubelet.service.d/` répertoire avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs de votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

`kubelet`configuration à partir des données utilisateur

Le `kubelet.service.d` répertoire doit être créé pour ce fichier. Vous devrez recharger systemd pour récupérer le fichier de configuration sans redémarrer. En AL2 023, le service sera probablement déjà en cours d'exécution lorsque votre script s'exécutera. Vous devrez donc également le redémarrer.

```
mkdir -p /etc/systemd/system/kubelet.service.d
echo '[Service]' > /etc/systemd/system/kubelet.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/kubelet.service.d/http-
proxy.conf
systemctl daemon-reload
systemctl restart kubelet
```

`ssm`configuration du proxy

`ssm`est l'un des fournisseurs d'informations d'identification qui peuvent être utilisés pour initialiser un nœud hybride. `ssm`est chargé de l'authentification AWS et de la génération des informations d'identification temporaires utilisées par`kubelet`. Si vous utilisez un proxy dans votre environnement sur site et que vous l'utilisez `ssm` comme fournisseur d'informations d'identification sur le nœud, vous devez le configurer `ssm` afin qu'il puisse communiquer avec les points de terminaison du service Amazon SSM.

Créez un fichier sur chaque nœud hybride appelé `http-proxy.conf` dans le chemin ci-dessous en fonction du système d'exploitation

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d/http-proxy.conf`
- Amazon Linux 2023 et Red Hat Enterprise Linux - `/etc/systemd/system/amazon-ssm-agent.service.d/http-proxy.conf`

Remplissez le fichier avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs de votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
```

```
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

ssmconfiguration à partir des données utilisateur

Le répertoire des fichiers de service `ssm` systemd doit être créé pour ce fichier. Le chemin du répertoire dépend du système d'exploitation utilisé sur le nœud.

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d`
- Amazon Linux 2023 et Red Hat Enterprise Linux - `/etc/systemd/system/amazon-ssm-agent.service.d`

Remplacez le nom du service systemd dans la commande de redémarrage ci-dessous en fonction du système d'exploitation utilisé sur le nœud

- Ubuntu - `snap.amazon-ssm-agent.amazon-ssm-agent`
- Amazon Linux 2023 et Red Hat Enterprise Linux - `amazon-ssm-agent`

```
mkdir -p systemd-service-file-directory
echo '[Service]' > [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://[.replaceable]#proxy-domain:port"' >> systemd-
service-file-directory/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://[.replaceable]#proxy-domain:port"' >>
 [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> [.replaceable]#systemd-service-file-
directory/http-proxy.conf
systemctl daemon-reload
systemctl restart [.replaceable]#systemd-service-name
```

Configuration du proxy du système d'exploitation

Si vous utilisez un proxy pour accéder à Internet, vous devez configurer votre système d'exploitation pour pouvoir extraire les dépendances des nœuds hybrides du gestionnaire de packages de votre système d'exploitation.

Ubuntu

1. Configurez `snap` pour utiliser votre proxy avec les commandes suivantes :

```
sudo snap set system proxy.https=http://proxy-domain:port
sudo snap set system proxy.http=http://proxy-domain:port
```

2. Pour activer le proxy pour`apt`, créez un fichier appelé `apt.conf` dans le `/etc/apt/` répertoire. Remplacez le domaine proxy et le port par les valeurs de votre environnement.

```
Acquire::http::Proxy "http://proxy-domain:port";
Acquire::https::Proxy "http://proxy-domain:port";
```

Amazon Linux 2023

1. Configurez `dnf` pour utiliser votre proxy. Créez un fichier `/etc/dnf/dnf.conf` contenant le domaine proxy et les valeurs de port de votre environnement.

```
proxy=http://proxy-domain:port
```

Utilisation de Red Hat Enterprise Linux

1. Configurez `yum` pour utiliser votre proxy. Créez un fichier `/etc/yum.conf` contenant le domaine proxy et les valeurs de port de votre environnement.

```
proxy=http://proxy-domain:port
```

Configuration du proxy IAM Roles Anywhere

Le service de fournisseur d'informations d'identification IAM Roles Anywhere est chargé d'actualiser les informations d'identification lorsque vous utilisez IAM Roles Anywhere avec le `enableCredentialsFile` drapeau (voir). the section called “Agent d'identité du pod EKS” Si vous utilisez un proxy dans votre environnement sur site, vous devez configurer le service afin qu'il puisse communiquer avec les points de terminaison IAM Roles Anywhere.

Créez un fichier appelé `http-proxy.conf` dans le `/etc/systemd/system/aws_signing_helper_update.service.d/` répertoire avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs de votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuration à l'échelle du cluster

Les configurations de cette section doivent être appliquées après avoir créé votre cluster Amazon EKS et avant de l'exécuter `nodeadm init` sur chaque nœud hybride.

configuration du proxy kube-proxy

Amazon EKS s'installe automatiquement `kube-proxy` sur chaque nœud hybride DaemonSet lorsque vos nœuds hybrides rejoignent le cluster. `kube-proxy`permet le routage entre les services soutenus par des pods sur les clusters Amazon EKS. Pour configurer chaque hôte, `kube-proxy` une résolution DNS est requise pour le point de terminaison de votre cluster Amazon EKS.

1. Modifiez le `kube-proxy` DaemonSet avec la commande suivante

   ```
   kubectl -n kube-system edit ds kube-proxy
   ```

   Cela ouvrira la `kube-proxy` DaemonSet définition dans l'éditeur que vous avez configuré.

2. Ajoutez les variables d'environnement pour `HTTP_PROXY` et`HTTPS_PROXY`. Notez que la variable d'`NODE_NAME`environnement doit déjà exister dans votre configuration. Remplacez `proxy-domain` et `port` par des valeurs pour votre environnement.

   ```
   containers:
     - command:
       - kube-proxy
       - --v=2
       - --config=/var/lib/kube-proxy-config/config - --hostname-override=$(NODE_NAME)
       env:
       - name: HTTP_PROXY
         value: http://proxy-domain:port
       - name: HTTPS_PROXY
         value: http://proxy-domain:port
   ```

```
    - name: NODE_NAME
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: spec.nodeName
```

Concepts pour les nœuds hybrides

Avec Amazon EKS Hybrid Nodes, vous associez des machines physiques ou virtuelles exécutées dans des environnements locaux ou périphériques à des clusters Amazon EKS exécutés dans le AWS cloud. Cette approche présente de nombreux avantages, mais introduit également de nouveaux concepts et architectures réseau pour ceux qui sont habitués à exécuter des clusters Kubernetes dans un environnement réseau unique.

Les sections suivantes examinent en détail les concepts de Kubernetes et de mise en réseau pour les nœuds hybrides EKS et détaillent la manière dont le trafic circule dans l'architecture hybride. Ces sections nécessitent que vous soyez familiarisé avec les connaissances de base du réseau Kubernetes, telles que les concepts de pods, de nœuds, de services, de plan de contrôle Kubernetes, de kubelet et de kube-proxy.

Nous vous recommandons de lire ces pages dans l'ordre, en commençant par lethe section called "Concepts de réseau", puis lethe section called "Concepts de Kubernetes", et enfin lethe section called "Flux de trafic".

Rubriques

- Concepts de mise en réseau pour les nœuds hybrides
- Concepts Kubernetes pour les nœuds hybrides
- Flux de trafic réseau pour les nœuds hybrides

Concepts de mise en réseau pour les nœuds hybrides

Cette section détaille les principaux concepts de réseau et les contraintes que vous devez prendre en compte lors de la conception de la topologie de votre réseau pour les nœuds hybrides EKS.

Concepts de mise en réseau pour les nœuds hybrides EKS

Le VPC en tant que hub réseau

Tout le trafic qui traverse les limites du cloud passe par votre VPC. Cela inclut le trafic entre le plan de contrôle EKS ou les pods s'exécutant AWS vers les nœuds hybrides ou les pods exécutés sur ceux-ci. Vous pouvez considérer le VPC de votre cluster comme le hub réseau entre vos nœuds hybrides et le reste du cluster. Cette architecture vous donne le contrôle total du trafic et de son routage, mais vous oblige également à configurer correctement les itinéraires, les groupes de sécurité et les pare-feux pour le VPC.

Plan de contrôle EKS vers le VPC

Le plan de contrôle EKS attache les interfaces réseau élastiques (ENIs) à votre VPC. Ils ENIs gèrent le trafic à destination et en provenance du serveur API EKS. Vous contrôlez le placement du plan de contrôle EKS ENIs lorsque vous configurez votre cluster, car EKS s'attache ENIs aux sous-réseaux que vous transmettez lors de la création du cluster.

EKS associe les groupes de sécurité aux groupes ENIs qu'EKS attache à vos sous-réseaux. Ces groupes de sécurité autorisent le trafic à destination et en provenance du plan de contrôle EKS via le

ENIs. Ceci est important pour les nœuds hybrides EKS, car vous devez autoriser le trafic provenant des nœuds hybrides et des pods qui s'y exécutent vers le plan de contrôle EKS ENIs.

Réseaux de nœuds distants

Les réseaux de nœuds distants, en particulier le nœud distant CIDRs, sont les plages de nœuds IPs attribuées aux machines que vous utilisez en tant que nœuds hybrides. Lorsque vous provisionnez des nœuds hybrides, ils résident dans votre centre de données sur site ou dans un emplacement périphérique, qui est un domaine réseau différent de celui du plan de contrôle EKS et du VPC. Chaque nœud hybride possède une ou plusieurs adresses IP provenant d'un nœud distant CIDR distinct des sous-réseaux de votre VPC.

Vous configurez le cluster EKS avec ces nœuds distants CIDRs afin qu'EKS sache qu'il doit acheminer tout le trafic destiné aux nœuds hybrides IPs via le VPC de votre cluster, comme les demandes adressées à l'API kubelet.

Réseaux Remote Pod

Les réseaux de pods distants sont les plages IPs attribuées aux pods exécutés sur les nœuds hybrides. Généralement, vous configurez votre CNI avec ces plages et la fonctionnalité de gestion des adresses IP (IPAM) du CNI se charge d'attribuer une tranche de ces plages à chaque nœud hybride. Lorsque vous créez un pod, le CNI attribue une adresse IP au pod à partir de la tranche allouée au nœud où le pod a été planifié.

Vous configurez le cluster EKS avec ces pods distants CIDRs afin que le plan de contrôle EKS sache qu'il doit acheminer tout le trafic destiné aux pods exécutés sur les nœuds hybrides via le VPC de votre cluster, par exemple pour les communications avec les webhooks.

Sur site, sur le VPC

Le réseau sur site que vous utilisez pour les nœuds hybrides doit être acheminé vers le VPC que vous utilisez pour votre cluster EKS. Plusieurs options de connectivité Network-to-Amazon VPC sont disponibles pour connecter votre réseau local à un VPC. Vous pouvez également utiliser votre propre solution VPN.

Il est important de configurer correctement le routage côté AWS cloud dans le VPC et dans votre réseau sur site, afin que les deux réseaux acheminent le trafic approprié via la connexion des deux réseaux.

Dans le VPC, tout le trafic destiné aux réseaux de nœuds distants et de pods distants doit être acheminé via la connexion vers votre réseau local (appelé « passerelle »). Si certains de vos sous-réseaux ont des tables de routage différentes, vous devez configurer chaque table de routage avec les routes pour vos nœuds hybrides et les pods qui y sont exécutés. Cela est vrai pour les sous-réseaux auxquels le plan ENIs de contrôle EKS est attaché, ainsi que pour les sous-réseaux contenant des EC2 nœuds ou des pods devant communiquer avec des nœuds hybrides.

Dans votre réseau sur site, vous devez configurer votre réseau pour autoriser le trafic à destination et en provenance du VPC de votre cluster EKS et les AWS autres services requis pour les nœuds hybrides. Le trafic du cluster EKS traverse la passerelle dans les deux directions.

Contraintes liées au réseau

Réseau entièrement routé

La principale contrainte est que le plan de contrôle EKS et tous les nœuds, qu'ils soient cloud ou hybrides, doivent former un réseau entièrement routé. Cela signifie que tous les nœuds doivent pouvoir se joindre les uns aux autres au niveau de la troisième couche, par adresse IP.

Le plan de contrôle EKS et les nœuds cloud sont déjà accessibles l'un depuis l'autre car ils se trouvent dans un réseau plat (le VPC). Les nœuds hybrides se trouvent toutefois dans un domaine de réseau différent. C'est pourquoi vous devez configurer un routage supplémentaire dans le VPC et sur votre réseau local pour acheminer le trafic entre les nœuds hybrides et le reste du cluster. Si les nœuds hybrides sont accessibles entre eux et depuis le VPC, vos nœuds hybrides peuvent se trouver dans un seul réseau plat ou dans plusieurs réseaux segmentés.

Module de télécommande routable CIDRs

Pour que le plan de contrôle EKS communique avec les pods exécutés sur des nœuds hybrides (par exemple, les webhooks ou le serveur Metrics) ou pour que les pods exécutés sur des nœuds cloud communiquent avec des pods exécutés sur des nœuds hybrides (communication est-ouest de la

charge de travail), le CIDR de votre pod distant doit être routable depuis le VPC. Cela signifie que le VPC doit être en mesure d'acheminer le trafic vers le pod CIDRs via la passerelle vers votre réseau local et que votre réseau local doit être en mesure d'acheminer le trafic d'un pod vers le bon nœud.

Il est important de noter la distinction entre les exigences de routage des pods dans le VPC et sur site. Le VPC doit uniquement savoir que tout trafic destiné à un pod distant doit passer par la passerelle. Si vous ne disposez que d'un seul modem CIDR, vous n'avez besoin que d'un seul itinéraire.

Cette exigence s'applique à tous les sauts de votre réseau sur site jusqu'au routeur local situé dans le même sous-réseau que vos nœuds hybrides. Il s'agit du seul routeur qui doit connaître la tranche CIDR du pod attribuée à chaque nœud, afin de s'assurer que le trafic d'un pod particulier est acheminé vers le nœud où le pod a été planifié.

Vous pouvez choisir de propager ces routes pour le pod local CIDRs depuis votre routeur local vers les tables de routage VPC, mais cela n'est pas nécessaire. Si votre espace sur site CIDRs change fréquemment et que vos tables de routage VPC doivent être mises à jour pour refléter le changement d' CIDRsespace, nous vous recommandons de propager l'espace sur site CIDRs vers les tables de routage VPC, mais cela est rare.

Notez que la contrainte permettant de rendre votre module local CIDRs routable est facultative. Si vous n'avez pas besoin d'exécuter des webhooks sur vos nœuds hybrides ou de laisser les pods sur les nœuds cloud communiquer avec les pods sur les nœuds hybrides, vous n'avez pas besoin de configurer le routage pour le pod CIDRs sur votre réseau local.

Pourquoi le pod sur site CIDRs doit-il être routable avec des nœuds hybrides ?

Lorsque vous utilisez EKS avec le VPC CNI pour vos nœuds de cloud, le VPC CNI est attribué directement IPs du VPC aux pods. Cela signifie qu'aucun routage spécial n'est nécessaire, car les modules cloud et le plan de contrôle EKS peuvent accéder IPs directement au pod.

Lorsqu'ils sont exécutés sur site (et avec d'autres CNIs dans le cloud), les pods s'exécutent généralement sur un réseau superposé isolé et le CNI se charge de distribuer le trafic entre les pods. Cela se fait généralement par le biais de l'encapsulation : le CNI convertit le pod-to-pod trafic en node-to-node trafic, en se chargeant de l'encapsulation et du désencapsulage des deux côtés. Ainsi, aucune configuration supplémentaire n'est nécessaire sur les nœuds et les routeurs.

La mise en réseau avec des nœuds hybrides est unique car elle combine les deux topologies : le plan de contrôle EKS et les nœuds cloud (avec le VPC CNI) supposent un réseau plat comprenant des nœuds et des pods, tandis que les pods exécutés sur des nœuds hybrides se trouvent dans un

réseau superposé utilisant VXLAN pour l'encapsulation (par défaut dans Cilium). Les pods exécutés sur des nœuds hybrides peuvent atteindre le plan de contrôle EKS et les pods exécutés sur des nœuds cloud en supposant que le réseau sur site puisse être acheminé vers le VPC. Toutefois, sans routage pour le module CIDRs sur le réseau local, tout trafic revenant vers une adresse IP d'espace local sera finalement abandonné si le réseau ne sait pas comment atteindre le réseau superposé et acheminer vers les nœuds appropriés.

Concepts Kubernetes pour les nœuds hybrides

Cette page détaille les concepts clés de Kubernetes qui sous-tendent l'architecture du système EKS Hybrid Nodes.

Plan de contrôle EKS dans le VPC

Les IPs du plan de contrôle EKS ENIs sont stockés dans l'`kubernetesEndpoints`objet dans l'espace de `default` noms. Lorsqu'EKS en crée de nouveaux ENIs ou en supprime d'anciens, EKS met à jour cet objet afin que la liste IPs soit toujours la même up-to-date.

Vous pouvez utiliser ces points de terminaison via le `kubernetes` Service, également dans l'espace de `default` noms. Ce `ClusterIP` type de service se voit toujours attribuer la première adresse IP du service CIDR du cluster. Par exemple, pour le service CIDR`172.16.0.0/16`, l'adresse IP du service sera`172.16.0.1`.

En général, c'est ainsi que les pods (qu'ils s'exécutent dans le cloud ou sur des nœuds hybrides) accèdent au serveur d'API EKS Kubernetes. Les pods utilisent l'adresse IP du service comme adresse IP de destination, qui est traduite en adresse réelle IPs de l'un des plans de contrôle EKS ENIs. La principale exception est `kube-proxy` que cela configure la traduction.

Point de terminaison du serveur API EKS

L'adresse IP du `kubernetes` service n'est pas le seul moyen d'accéder au serveur API EKS. EKS crée également un nom DNS Route53 lorsque vous créez votre cluster. Il s'agit du `endpoint` champ de votre cluster EKS lorsque vous appelez l'action `DescribeCluster` API EKS.

```
{
    "cluster": {
        "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
        "name": "my-cluster",
        "status": "ACTIVE"
    }
```

```
}
```

Dans un cluster d'accès aux terminaux publics ou dans un cluster d'accès aux points de terminaison publics et privés, vos nœuds hybrides transformeront ce nom DNS en une adresse IP publique par défaut, routable via Internet. Dans un cluster d'accès aux terminaux privés, le nom DNS devient le nom privé IPs du plan de contrôle EKS ENIs.

C'est ainsi que `kubelet` et `kube-proxy` accèdent au serveur d'API Kubernetes. Si vous souhaitez que tout le trafic de votre cluster Kubernetes passe par le VPC, vous devez soit configurer votre cluster en mode d'accès privé, soit modifier votre serveur DNS sur site pour transformer le point de terminaison du cluster EKS en point privé IPs du plan de contrôle EKS. ENIs

Point de terminaison **`kubelet`**

Cela `kubelet` expose plusieurs points de terminaison REST, permettant à d'autres parties du système d'interagir avec chaque nœud et de recueillir des informations à partir de celui-ci. Dans la plupart des clusters, la majeure partie du trafic vers le `kubelet` serveur provient du plan de contrôle, mais certains agents de surveillance peuvent également interagir avec celui-ci.

Grâce à cette interface, il `kubelet` gère diverses requêtes : récupération des journaux (`kubectl logs`), exécution de commandes dans des conteneurs (`kubectl exec`) et transfert de trafic (`kubectl port-forward`). Chacune de ces demandes interagit avec le runtime du conteneur sous-jacent via le`kubelet`, ce qui apparaît parfaitement aux administrateurs et aux développeurs du cluster.

Le client le plus courant de cette API est le serveur d'API Kubernetes. Lorsque vous utilisez l'une des `kubectl` commandes mentionnées précédemment, `kubectl` envoyez une demande d'API au serveur d'API, qui appelle ensuite l'`kubelet`API du nœud sur lequel le pod est exécuté. C'est la principale raison pour laquelle l'adresse IP du nœud doit être accessible depuis le plan de contrôle EKS et pourquoi, même si vos pods sont en cours d'exécution, vous ne pourrez pas accéder à leurs journaux ou `exec` si la route du nœud est mal configurée.

Nœud IPs

Lorsque le plan de contrôle EKS communique avec un nœud, il utilise l'une des adresses indiquées dans le statut de l'`Node`objet (`status.addresses`).

Avec les nœuds cloud EKS, il est courant que le kubelet indique l'adresse IP privée de l' EC2 instance `InternalIP` lors de l'enregistrement du nœud. Cette adresse IP est ensuite validée par le Cloud Controller Manager (CCM) pour s'assurer qu'elle appartient à l' EC2

instance. En outre, le CCM ajoute généralement les noms public IPs (as`ExternalIP`) et DNS (`InternalDNS`et`ExternalDNS`) de l'instance à l'état du nœud.

Cependant, il n'existe pas de CCM pour les nœuds hybrides. Lorsque vous enregistrez un nœud hybride avec la CLI EKS Hybrid Nodes (`nodeadm`), celui-ci configure le kubelet pour qu'il indique l'adresse IP de votre machine directement dans l'état du nœud, sans le CCM.

```
apiVersion: v1
kind: Node
metadata:
  name: my-node-1
spec:
  providerID: eks-hybrid:///us-west-2/my-cluster/my-node-1
status:
  addresses:
  - address: 10.1.1.236
    type: InternalIP
  - address: my-node-1
    type: Hostname
```

Si votre machine en possède plusieurs IPs, le kubelet sélectionnera l'un d'entre eux selon sa propre logique. Vous pouvez contrôler l'adresse IP sélectionnée à l'aide de l'`--node-ip`indicateur, que vous pouvez transmettre dans la `nodeadm` configuration`spec.kubelet.flags`. Seule l'adresse IP indiquée dans l'`Node`objet a besoin d'une route depuis le VPC. Vos machines peuvent en avoir d'autres IPs qui ne sont pas accessibles depuis le cloud.

`kube-proxy`

`kube-proxy`est responsable de la mise en œuvre de l'abstraction du service au niveau de la couche réseau de chaque nœud. Il agit comme un proxy réseau et un équilibreur de charge pour le trafic destiné aux services Kubernetes. En surveillant en permanence le serveur d'API Kubernetes pour détecter les modifications liées aux services et aux points de terminaison, il met à jour de `kube-proxy` manière dynamique les règles réseau de l'hôte sous-jacent afin de garantir que le trafic est correctement dirigé.

En `iptables` mode, `kube-proxy` programme plusieurs `netfilter` chaînes pour gérer le trafic de service. Les règles forment la hiérarchie suivante :

1. Chaîne KUBE-SERVICES : point d'entrée pour tout le trafic de services. Il a des règles correspondant à chaque service `ClusterIP` et à chaque port.

2. KUBE-SVC-XXX chaînes : les chaînes spécifiques aux services ont des règles d'équilibrage de charge pour chaque service.
3. KUBE-SEP-XXX chaînes : les chaînes spécifiques aux points de terminaison ont les règles réelles. `DNAT`

Examinons ce qui se passe pour un service `test-server` dans l'espace de `default` noms :* Service ClusterIP : * Port du service :* `172.16.31.14` Backing pods `80` :, et `10.2.0.110` `10.2.1.39` `10.2.2.254`

Lorsque nous inspectons les `iptables` règles (en utilisant`iptables-save 0 grep -A10 KUBE-SERVICES`) :

1. Dans la chaîne KUBE-SERVICES, nous trouvons une règle correspondant au service :

```
-A KUBE-SERVICES -d 172.16.31.14/32 -p tcp -m comment --comment "default/test-server
 cluster IP" -m tcp --dport 80 -j KUBE-SVC-XYZABC123456
```

   - Cette règle fait correspondre les paquets destinés au 172.16.31. 14:80
   - Le commentaire indique à quoi sert cette règle : `default/test-server cluster IP`
   - Les paquets correspondants passent à la `KUBE-SVC-XYZABC123456` chaîne
2. La XYZABC123456 chaîne KUBE-SVC- possède des règles d'équilibrage de charge basées sur les probabilités :

```
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.33333333349 -j
 KUBE-SEP-POD1XYZABC
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.50000000000 -j
 KUBE-SEP-POD2XYZABC
-A KUBE-SVC-XYZABC123456 -j KUBE-SEP-POD3XYZABC
```

   - Première règle : 33,3 % de chances de passer à `KUBE-SEP-POD1XYZABC`
   - Deuxième règle : 50 % de chances que le trafic restant (33,3 % du total) passe à `KUBE-SEP-POD2XYZABC`
   - Dernière règle : tout le trafic restant (33,3 % du total) passe à `KUBE-SEP-POD3XYZABC`
3. Les chaînes KUBE-SEP-XXX individuelles exécutent le DNAT (Destination NAT) :

```
-A KUBE-SEP-POD1XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.0.110:80
-A KUBE-SEP-POD2XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.1.39:80
```

```
-A KUBE-SEP-POD3XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.2.254:80
```

- Ces règles DNAT réécrivent l'adresse IP et le port de destination pour diriger le trafic vers des pods spécifiques.
- Chaque règle gère environ 33,3 % du trafic, assurant un équilibrage de charge uniforme entre`10.2.0.110`, `10.2.1.39` et`10.2.2.254`.

Cette structure de chaîne à plusieurs niveaux permet d'`kube-proxy`implémenter efficacement l'équilibrage de charge de service et la redirection par le biais de la manipulation de paquets au niveau du noyau, sans nécessiter de processus proxy dans le chemin de données.

Impact sur les opérations Kubernetes

Une panne `kube-proxy` sur un nœud empêche ce nœud de router correctement le trafic de service, ce qui entraîne des délais d'attente ou des échecs de connexion pour les pods qui dépendent des services du cluster. Cela peut être particulièrement perturbateur lorsqu'un nœud est enregistré pour la première fois. Le CNI doit communiquer avec le serveur d'API Kubernetes pour obtenir des informations, telles que le CIDR du pod du nœud, avant de pouvoir configurer le réseau d'un pod. Pour ce faire, il utilise l'adresse IP du `kubernetes` service. Toutefois, si vous `kube-proxy` n'avez pas pu démarrer ou si vous n'avez pas défini les bonnes `iptables` règles, les demandes envoyées à l'adresse IP du `kubernetes` service ne sont pas traduites en données réelles sur le plan IPs de contrôle EKS ENIs. En conséquence, le CNI entrera dans une boucle de crash et aucun des pods ne pourra fonctionner correctement.

Nous savons que les pods utilisent l'adresse IP du `kubernetes` service pour communiquer avec le serveur d'API Kubernetes, mais ils `kube-proxy` doivent d'abord définir des `iptables` règles pour que cela fonctionne.

Comment `kube-proxy` communique-t-on avec le serveur API ?

Ils `kube-proxy` doivent être configurés pour utiliser le véritable nom IP/s du serveur d'API Kubernetes ou un nom DNS qui correspond à eux. Dans le cas d'EKS, EKS configure la valeur par défaut `kube-proxy` pour qu'elle pointe vers le nom DNS Route53 créé par EKS lorsque vous créez le cluster. Vous pouvez voir cette valeur `kube-proxy` ConfigMap dans le `kube-system` namespace. Le contenu de `kubeconfig` ce produit ConfigMap est injecté dans le `kube-proxy` pod, alors recherchez le `clusters0.cluster.server` champ. Cette valeur correspondra au `endpoint` champ de votre cluster EKS (lorsque vous appelez `DescribeCluster` l'API EKS).

```
apiVersion: v1
```

```
data:
  kubeconfig: |-
    kind: Config
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        server: https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com
      name: default
    contexts:
    - context:
        cluster: default
        namespace: default
        user: default
      name: default
    current-context: default
    users:
    - name: default
      user:
        tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

Pod à distance routable CIDRs

La the section called "Concepts de réseau" page détaille les conditions requises pour exécuter des webhooks sur des nœuds hybrides ou pour que des pods exécutés sur des nœuds cloud communiquent avec des pods exécutés sur des nœuds hybrides. La principale exigence est que le routeur sur site sache quel nœud est responsable de l'adresse IP d'un pod en particulier. Il existe plusieurs moyens d'y parvenir, notamment le Border Gateway Protocol (BGP), les routes statiques et le proxy ARP (Address Resolution Protocol). Elles sont abordées dans les sections suivantes.

Protocole BGP (Border Gateway Protocol)

Si votre CNI le prend en charge (comme Cilium et Calico), vous pouvez utiliser le mode BGP de votre CNI pour propager les routes vers votre pod par nœud CIDRs depuis vos nœuds vers votre routeur local. Lorsque vous utilisez le mode BGP du CNI, votre CNI agit comme un routeur virtuel. Votre routeur local pense donc que le pod CIDR appartient à un sous-réseau différent et que votre nœud est la passerelle vers ce sous-réseau.

On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
BGP UPDATE from 10.80.0.2 to Router:
ANNOUNCE:
10.85.1.0/24 via 10.80.0.2
BGP UPDATE from 10.80.0.3 to Router:
ANNOUNCE:
10.85.2.0/24 via 10.80.0.3
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [dynamic] via 10.80.0.2 (BGP)
- 10.85.2.0/24 [dynamic] via 10.80.0.3 (BGP)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

Routes statiques

Vous pouvez également configurer des itinéraires statiques dans votre routeur local. C'est le moyen le plus simple d'acheminer le code CIDR du pod sur site vers votre VPC, mais c'est également le moyen le plus sujet aux erreurs et le plus difficile à gérer. Vous devez vous assurer que les itinéraires utilisent toujours up-to-date les nœuds existants et le pod qui leur est attribué CIDRs. Si votre nombre de nœuds est faible et que l'infrastructure est statique, il s'agit d'une option viable qui élimine le besoin de support BGP dans votre routeur. Si vous optez pour cette solution, nous vous recommandons de configurer votre CNI avec la tranche CIDR du pod que vous souhaitez attribuer à chaque nœud au lieu de laisser son IPAM décider.

On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [static] via 10.80.0.2 (STATIC)
- 10.85.2.0/24 [static] via 10.80.0.3 (STATIC)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

Proxy du protocole de résolution d'adresses (ARP)

Le proxy ARP est une autre approche permettant de rendre les pods sur site IPs routables, particulièrement utile lorsque vos nœuds hybrides se trouvent sur le même réseau de couche 2 que votre routeur local. Lorsque le proxy ARP est activé, un nœud répond aux demandes ARP pour le pod IPs qu'il héberge, même si celui-ci IPs appartient à un sous-réseau différent.

Lorsqu'un appareil de votre réseau local essaie d'atteindre l'adresse IP d'un pod, il envoie d'abord une demande ARP demandant « Qui possède cette adresse IP ? ». Le nœud hybride hébergeant ce pod répondra avec sa propre adresse MAC en disant « Je peux gérer le trafic pour cette adresse IP ». Cela crée un chemin direct entre les appareils de votre réseau local et les pods sans nécessiter de configuration de routeur.

Pour que cela fonctionne, votre CNI doit prendre en charge la fonctionnalité ARP du proxy. Cilium dispose d'un support intégré pour le proxy ARP que vous pouvez activer via la configuration. La principale considération est que le CIDR du pod ne doit pas se chevaucher avec un autre réseau de votre environnement, car cela pourrait entraîner des conflits de routage.

Cette approche présente plusieurs avantages :* Pas besoin de configurer votre routeur avec BGP ou de maintenir des routes statiques * Fonctionne bien dans les environnements où vous n'avez aucun contrôle sur la configuration de votre routeur

Pod-to-Pod encapsulation

Dans les environnements sur site, utilisez CNIs généralement des protocoles d'encapsulation pour créer des réseaux superposés capables de fonctionner au-dessus du réseau physique sans qu'il soit nécessaire de le reconfigurer. Cette section explique le fonctionnement de cette encapsulation. Notez que certains détails peuvent varier en fonction du CNI que vous utilisez.

L'encapsulation enveloppe les paquets réseau du pod d'origine dans un autre paquet réseau qui peut être acheminé via le réseau physique sous-jacent. Cela permet aux pods de communiquer entre des nœuds exécutant le même CNI sans que le réseau physique sache comment acheminer ces pods CIDRs.

Le protocole d'encapsulation le plus couramment utilisé avec Kubernetes est le Virtual Extensible LAN (VXLAN), bien que d'autres (tels que`Geneve`) soient également disponibles en fonction de votre CNI.

Encapsulation VXLAN

VXLAN encapsule les trames Ethernet de couche 2 dans des paquets UDP. Lorsqu'un pod envoie du trafic vers un autre pod sur un autre nœud, le CNI effectue les opérations suivantes :

1. Le CNI intercepte les paquets du Pod A
2. Le CNI enveloppe le paquet d'origine dans un en-tête VXLAN
3. Ce paquet encapsulé est ensuite envoyé via la pile réseau normale du nœud vers le nœud de destination
4. Le CNI du nœud de destination déballe le paquet et le livre au Pod B

Voici ce qui arrive à la structure des paquets lors de l'encapsulation VXLAN :

Emballage d' Pod-to-Podorigine :

```
+-----------------+---------------+-------------+-----------------+
| Ethernet Header | IP Header     | TCP/UDP     | Payload         |
| Src: Pod A MAC  | Src: Pod A IP | Src Port    |                 |
| Dst: Pod B MAC  | Dst: Pod B IP | Dst Port    |                 |
+-----------------+---------------+-------------+-----------------+
```

Après l'encapsulation VXLAN :

```
+-----------------+-------------+--------------+------------
+---------------------------+
| Outer Ethernet  | Outer IP    | Outer UDP    | VXLAN      | Original Pod-to-Pod
 |
| Src: Node A MAC | Src: Node A | Src: Random  | VNI: xx    | Packet (unchanged
 |
| Dst: Node B MAC | Dst: Node B | Dst: 4789    |            | from above)
 |
```

```
+-----------------+-------------+--------------+------------
+---------------------------+
```

L'identifiant de réseau VXLAN (VNI) fait la distinction entre les différents réseaux superposés.

Scénarios de communication du pod

Pods sur le même nœud hybride

Lorsque les pods d'un même nœud hybride communiquent, aucune encapsulation n'est généralement nécessaire. Le CNI met en place des routes locales qui dirigent le trafic entre les pods via les interfaces virtuelles internes du nœud :

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

Le paquet ne quitte jamais le nœud et ne nécessite pas d'encapsulation.

Pods sur différents nœuds hybrides

La communication entre les pods sur différents nœuds hybrides nécessite une encapsulation :

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B
 network -> [VXLAN decapsulation] -> CNI -> Pod B
```

Cela permet au trafic du pod de traverser l'infrastructure réseau physique sans que le réseau physique doive comprendre le routage IP du pod.

Flux de trafic réseau pour les nœuds hybrides

Cette page détaille les flux de trafic réseau pour les nœuds hybrides EKS avec des diagrammes montrant les chemins end-to-end réseau pour les différents types de trafic.

Les flux de trafic suivants sont couverts :

- the section called "Nœud hybride `kubelet` vers plan de contrôle EKS"
- the section called "Plan de contrôle EKS vers nœud hybride (`kubelet`serveur)"
- the section called "Pods exécutés sur des nœuds hybrides vers le plan de contrôle EKS"
- the section called "Plan de contrôle EKS pour les pods exécutés sur un nœud hybride (webhooks)"
- the section called "Pod-to-Pod exécution sur des nœuds hybrides"

- the section called "Des pods sur des nœuds cloud vers des pods sur des nœuds hybrides (trafic est-ouest)"

Nœud hybride **`kubelet`** vers plan de contrôle EKS

Demande

1**`kubelet`**. Lance une demande

Lorsque `kubelet` le nœud hybride doit communiquer avec le plan de contrôle EKS (par exemple, pour signaler l'état du nœud ou obtenir les spécifications du pod), il utilise le `kubeconfig` fichier fourni lors de l'enregistrement du nœud. Il `kubeconfig` contient l'URL du point de terminaison du serveur API (le nom DNS Route53) plutôt que des adresses IP directes.

`kubelet`Effectue une recherche DNS pour le point de terminaison (par exemple, `https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com`). Dans un cluster d'accès public, cela se résout en une adresse IP publique (par exemple`54.239.118.52`) qui

appartient au service EKS en cours d'exécution AWS. Il crée `kubelet` ensuite une requête HTTPS sécurisée vers ce point de terminaison. Le paquet initial ressemble à ceci :

```
+--------------------+---------------------+-----------------+
| IP Header          | TCP Header          | Payload         |
| Src: 10.80.0.2     | Src: 52390 (random) |                 |
| Dst: 54.239.118.52 | Dst: 443            |                 |
+--------------------+---------------------+-----------------+
```

2. Routage du routeur local

Étant donné que l'adresse IP de destination est une adresse IP publique qui ne fait pas partie du réseau local, `kubelet` envoie ce paquet à sa passerelle par défaut (le routeur local sur site). Le routeur examine l'adresse IP de destination et détermine qu'il s'agit d'une adresse IP publique.

Pour le trafic public, le routeur transmet généralement le paquet à une passerelle Internet ou à un routeur frontalier qui gère le trafic sortant vers Internet. Ceci est omis dans le schéma et dépend de la configuration de votre réseau local. Le paquet traverse votre infrastructure réseau sur site et finit par atteindre le réseau de votre fournisseur de services Internet.

3. Livraison au plan de commande EKS

Le paquet traverse l'Internet public et les réseaux de transit jusqu'à ce qu' AWS il atteigne le réseau. AWS le réseau achemine le paquet vers le point de terminaison du service EKS dans la région appropriée. Lorsque le paquet atteint le service EKS, il est transféré vers le plan de contrôle EKS réel de votre cluster.

Ce routage via l'Internet public est différent du chemin privé routé par VPC que nous verrons dans d'autres flux de trafic. La principale différence est que lorsque vous utilisez le mode d'accès public, le trafic en provenance des locaux `kubelet` (mais pas des pods) vers le plan de contrôle EKS ne passe pas par votre VPC. Il utilise plutôt l'infrastructure Internet mondiale.

Réponse

Une fois que le plan de contrôle EKS a traité la `kubelet` demande, il renvoie une réponse :

3. Le plan de contrôle EKS envoie une réponse

Le plan de contrôle EKS crée un paquet de réponse. Ce paquet a l'adresse IP publique comme source et l'adresse IP du nœud hybride comme destination :

```
+--------------------+---------------------+-----------------+
```

```
| IP Header          | TCP Header          | Payload         |
| Src: 54.239.118.52 | Src: 443            |                 |
| Dst: 10.80.0.2     | Dst: 52390          |                 |
+--------------------+---------------------+-----------------+
```

2. Routage Internet

Le paquet de réponse est renvoyé par Internet, en suivant le chemin de routage déterminé par les fournisseurs de services Internet, jusqu'à ce qu'il atteigne votre routeur périphérique de réseau local.

1. Livraison locale

Votre routeur local reçoit le paquet et reconnaît que l'adresse IP de destination (`10.80.0.2`) appartient à votre réseau local. Il transmet le paquet via votre infrastructure réseau locale jusqu'à ce qu'il atteigne le nœud hybride cible, où il `kubelet` reçoit et traite la réponse.

Nœud hybride **`kube-proxy`** vers plan de contrôle EKS

Si vous activez l'accès public aux points de terminaison pour le cluster, le trafic de retour utilise l'Internet public. Ce trafic provient du nœud hybride vers `kube-proxy` le plan de contrôle EKS et suit le même chemin que le trafic du nœud `kubelet` vers le plan de contrôle EKS.

Plan de contrôle EKS vers nœud hybride (**`kubelet`**serveur)

Demande

1. Le serveur d'API EKS Kubernetes lance une demande

Le serveur API EKS Kubernetes extrait l'adresse IP du nœud (`10.80.0.2`) à partir de l'état de l'objet nœud. Il achemine ensuite cette demande via son ENI dans le VPC, car l'adresse IP de destination appartient au nœud distant configuré CIDR (). `10.80.0.0/16` Le paquet initial ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 67493 (random) |                 |
| Dst: 10.80.0.2  | Dst: 10250          |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement du réseau VPC

Le paquet quitte l'ENI et entre dans la couche réseau VPC, où il est dirigé vers la passerelle du sous-réseau pour un routage ultérieur.

3. Recherche dans une table de routage VPC

La table de routage VPC du sous-réseau contenant le plan de contrôle EKS ENI possède une route spécifique (la deuxième du schéma) pour le nœud distant CIDR. Sur la base de cette règle de routage, le paquet est dirigé vers la VPC-to-onprem passerelle.

4. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau local.

5. Réception réseau sur site

Le paquet arrive à votre routeur local local qui gère le trafic pour le sous-réseau où se trouvent vos nœuds hybrides.

6. Livraison finale

Le routeur local identifie que l'adresse IP de destination (`10.80.0.2`) appartient à son réseau directement connecté et transmet le paquet directement au nœud hybride cible, où il `kubelet` reçoit et traite la demande.

Réponse

Une fois que le nœud hybride `kubelet` a traité la demande, il renvoie une réponse en suivant le même chemin en sens inverse :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 10250          |                 |
| Dst: 10.0.0.132 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

6. **`kubelet`** Envoie une réponse

`kubelet`Le nœud hybride (`10.80.0.2`) crée un paquet de réponse avec l'adresse IP source d'origine comme destination. La destination n'appartenant pas au réseau local, elle est envoyée à la passerelle par défaut de l'hôte, qui est le routeur local.

5. Routage du routeur local

Le routeur détermine l'adresse IP de destination (`10.0.0.132`) à `10.0.0.0/16` laquelle appartient une route pointant vers la passerelle de connexion à AWS.

4. Retour transfrontalier

Le paquet est renvoyé via la même connexion sur site vers un VPC (telle que Direct Connect ou VPN), franchissant la limite du cloud dans le sens inverse.

3. Routage VPC

Lorsque le paquet arrive dans le VPC, les tables de routage indiquent que l'adresse IP de destination appartient à un CIDR VPC. Les itinéraires de paquets au sein du VPC.

2. Livraison d'un réseau VPC

La couche réseau VPC transmet le paquet au sous-réseau avec le plan de contrôle EKS ENI (). `10.0.0.132`

1. Réception ENI

Le paquet atteint le plan de contrôle EKS ENI attaché au serveur d'API Kubernetes, achevant ainsi l'aller-retour.

Pods exécutés sur des nœuds hybrides vers le plan de contrôle EKS

Sans CNI NAT

Demande

Les pods communiquent généralement avec le serveur d'API Kubernetes via le service. `kubernetes` L'adresse IP du service est la première adresse IP du service CIDR du cluster. Cette convention permet aux pods qui doivent s'exécuter avant que CoreDNS ne soit disponible d'atteindre le serveur d'API, par exemple le CNI. Les demandes quittent le module avec l'adresse IP du service comme destination. Par exemple, si le CIDR du service est`172.16.0.0/16`, l'adresse IP du service sera`172.16.0.1`.

1. Le module lance une demande

Le pod envoie une demande à l'adresse IP du `kubernetes` service (`172.16.0.1`) sur le port du serveur API (443) à partir d'un port source aléatoire. Le paquet ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
```

```
+-----------------+---------------------+-----------------+
```

2. Traitement CNI

Le CNI détecte que l'adresse IP de destination n'appartient à aucun des pods CIDR qu'il gère. Le NAT sortant étant désactivé, le CNI transmet le paquet à la pile du réseau hôte sans le modifier.

3. Traitement du réseau de nœuds

Le paquet entre dans la pile réseau du nœud où les `netfilter` hooks déclenchent les `iptables` règles définies par kube-proxy. Plusieurs règles s'appliquent dans l'ordre suivant :

1. Le paquet atteint d'abord la `KUBE-SERVICES` chaîne, qui contient des règles correspondant au ClusterIP et au port de chaque service.
2. La règle correspondante passe à la `KUBE-SVC-XXX` chaîne du `kubernetes` service (paquets destinés à`172.16.0.1:443`), qui contient des règles d'équilibrage de charge.
3. La règle d'équilibrage de charge sélectionne aléatoirement l'une des `KUBE-SEP-XXX` chaînes pour le plan de commande ENI IPs (`10.0.0.132`or`10.0.1.23`).
4. La `KUBE-SEP-XXX` chaîne sélectionnée possède la règle réelle qui change l'adresse IP de destination de l'adresse IP du service à l'adresse IP sélectionnée. C'est ce qu'on appelle la traduction d'adresses réseau de destination (DNAT).

Une fois ces règles appliquées, en supposant que l'adresse IP du plan de contrôle EKS ENI sélectionné est l'adresse IP du plan de contrôle EKS sélectionné`10.0.0.132`, le paquet ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Le nœud transmet le paquet à sa passerelle par défaut car l'adresse IP de destination ne se trouve pas sur le réseau local.

4. Routage du routeur local

Le routeur local détermine que l'adresse IP de destination (`10.0.0.132`) appartient au VPC CIDR (`10.0.0.0/16`) et la transmet à la passerelle qui se connecte à. AWS

5. Transit transfrontalier

Le paquet passe par votre connexion établie (telle que Direct Connect ou VPN) à travers la limite du cloud jusqu'au VPC.

6. Livraison d'un réseau VPC

La couche réseau VPC achemine le paquet vers le sous-réseau approprié où se trouve le plan de contrôle EKS ENI (`10.0.0.132`).

7. Réception ENI

Le paquet atteint le plan de contrôle EKS ENI attaché au serveur API Kubernetes.

Réponse

Une fois que le plan de contrôle EKS a traité la demande, il renvoie une réponse au pod :

7. Le serveur API envoie une réponse

Le serveur d'API EKS Kubernetes crée un paquet de réponse avec l'adresse IP source d'origine comme destination. Le paquet ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Comme l'adresse IP de destination appartient au pod distant CIDR (`10.85.0.0/16`) configuré, il l'envoie via son ENI dans le VPC avec le routeur du sous-réseau comme saut suivant.

6. Routage VPC

La table de routage VPC contient une entrée pour le pod distant CIDR (`10.85.0.0/16`), qui dirige ce trafic vers la passerelle. VPC-to-onprem

5. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau local.

4. Réception réseau sur site

Le paquet arrive à votre routeur local sur site.

3. Livraison au nœud

La table du routeur contient une entrée pour `10.85.1.0/24` with `10.80.0.2` comme prochain saut, livrant le paquet à notre nœud.

2. Traitement du réseau de nœuds

Au fur et à mesure que le paquet est traité par la pile réseau du nœud, `conntrack` (une partie de`netfilter`) fait correspondre le paquet à la connexion initialement établie par le pod. Puisque le DNAT a été initialement appliqué, `conntrack` inverse le DNAT en réécrivant l'adresse IP source de l'adresse IP du plan de contrôle EKS ENI vers l'adresse IP du service : `kubernetes`

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Traitement CNI

Le CNI identifie que l'adresse IP de destination appartient à un pod de son réseau et envoie le paquet à l'espace de noms du réseau de pods approprié.

Ce flux montre pourquoi Remote Pod CIDRs doit être correctement routable depuis le VPC jusqu'au nœud spécifique hébergeant chaque pod. Le chemin de retour complet dépend du routage correct du pod sur les réseaux cloud et IPs sur site.

Avec CNI NAT

Ce flux est très similaire à celui sans CNI NAT, mais avec une différence essentielle : le CNI applique le NAT source (SNAT) au paquet avant de l'envoyer à la pile réseau du nœud. Cela remplace l'adresse IP source du paquet par l'adresse IP du nœud, ce qui permet au paquet d'être routé vers le nœud sans nécessiter de configuration de routage supplémentaire.

Demande

1. Le module lance une demande

Le pod envoie une demande à l'adresse IP du `kubernetes` service (`172.16.0.1`) sur le port du serveur d'API EKS Kubernetes (443) à partir d'un port source aléatoire. Le paquet ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement CNI

Le CNI détecte que l'adresse IP de destination n'appartient à aucun des pods CIDR qu'il gère. Le NAT sortant étant activé, le CNI applique le SNAT au paquet, en remplaçant l'IP source par l'IP du nœud avant de le transmettre à la pile réseau du nœud :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Remarque : CNI et CNI `iptables` sont présentés dans l'exemple sous forme de blocs séparés pour plus de clarté, mais dans la pratique, il est possible que certains CNIs utilisent le NAT `iptables` pour appliquer le NAT.

3. Traitement du réseau de nœuds

Ici, les `iptables` règles définies par `kube-proxy` se comportent de la même manière que dans l'exemple précédent, en équilibrant la charge du paquet sur l'un des plans de contrôle EKS ENIs. Le paquet ressemble maintenant à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Le nœud transmet le paquet à sa passerelle par défaut car l'adresse IP de destination ne se trouve pas sur le réseau local.

4. Routage du routeur local

Le routeur local détermine que l'adresse IP de destination (`10.0.0.132`) appartient au VPC CIDR (`10.0.0.0/16`) et la transmet à la passerelle qui se connecte à. AWS

5. Transit transfrontalier

Le paquet passe par votre connexion établie (telle que Direct Connect ou VPN) à travers la limite du cloud jusqu'au VPC.

6. Livraison d'un réseau VPC

La couche réseau VPC achemine le paquet vers le sous-réseau approprié où se trouve le plan de contrôle EKS ENI (`10.0.0.132`).

7. Réception ENI

Le paquet atteint le plan de contrôle EKS ENI attaché au serveur API Kubernetes.

Réponse

Une fois que le plan de contrôle EKS a traité la demande, il renvoie une réponse au pod :

7. Le serveur API envoie une réponse

Le serveur d'API EKS Kubernetes crée un paquet de réponse avec l'adresse IP source d'origine comme destination. Le paquet ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Comme l'adresse IP de destination appartient au nœud distant configuré CIDR (`10.80.0.0/16`), il l'envoie via son ENI dans le VPC avec le routeur du sous-réseau comme saut suivant.

6. Routage VPC

La table de routage VPC contient une entrée pour le nœud distant CIDR (`10.80.0.0/16`), qui dirige ce trafic vers la passerelle. VPC-to-onprem

5. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau local.

4. Réception réseau sur site

Le paquet arrive à votre routeur local sur site.

3. Livraison au nœud

Le routeur local identifie que l'adresse IP de destination (`10.80.0.2`) appartient à son réseau directement connecté et transmet le paquet directement au nœud hybride cible.

2. Traitement du réseau de nœuds

Lorsque le paquet est traité par la pile réseau du nœud, `conntrack` (une partie de`netfilter`) le met en correspondance avec la connexion initialement établie par le pod et, comme le DNAT a été initialement appliqué, il inverse cette tendance en réécrivant l'adresse IP source de l'adresse IP du plan de contrôle EKS ENI vers l'`kubernetes`adresse IP du service :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Traitement CNI

Le CNI identifie que ce paquet appartient à une connexion à laquelle il a précédemment appliqué le SNAT. Il inverse le SNAT, remplaçant l'adresse IP de destination par l'adresse IP du pod :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Le CNI détecte que l'adresse IP de destination appartient à un pod de son réseau et envoie le paquet à l'espace de noms du réseau de pods approprié.

Ce flux montre comment CNI Nat-ing peut simplifier la configuration en permettant le renvoi des paquets vers le nœud sans nécessiter de routage supplémentaire pour le pod. CIDRs

Plan de contrôle EKS pour les pods exécutés sur un nœud hybride (webhooks)

Ce schéma de trafic est le plus souvent observé avec les webhooks, où le plan de contrôle EKS doit établir directement des connexions aux serveurs webhooks exécutés dans des pods sur des nœuds hybrides. Les exemples incluent la validation et la mutation des webhooks d'admission, qui sont appelés par le serveur API lors des processus de validation ou de mutation des ressources.

Demande

1. Le serveur d'API EKS Kubernetes lance une demande

Lorsqu'un webhook est configuré dans le cluster et qu'une opération d'API pertinente le déclenche, le serveur d'API EKS Kubernetes doit établir une connexion directe avec le pod du serveur webhook. Le serveur API recherche d'abord l'adresse IP du pod à partir de la ressource Service ou Endpoint associée au webhook.

En supposant que le pod webhook s'exécute sur un nœud hybride doté d'une adresse IP`10.85.1.23`, le serveur d'API EKS Kubernetes crée une requête HTTPS à destination du point de terminaison du webhook. Le paquet initial est envoyé via le plan de contrôle EKS ENI de votre VPC car l'adresse IP de destination `10.85.1.23` appartient au module distant CIDR () configuré. `10.85.0.0/16` Le paquet ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 41892 (random) |                 |
| Dst: 10.85.1.23 | Dst: 8443           |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement réseau VPC

Le paquet quitte le plan de contrôle EKS ENI et entre dans la couche réseau VPC avec le routeur du sous-réseau comme saut suivant.

3. Recherche dans une table de routage VPC

La table de routage VPC du sous-réseau contenant le plan de contrôle EKS ENI contient un itinéraire spécifique pour le pod distant CIDR (). `10.85.0.0/16` Cette règle de routage dirige le paquet vers la VPC-to-onprem passerelle (par exemple, une passerelle privée virtuelle pour les connexions Direct Connect ou VPN) :

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

4. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau local. Le paquet conserve ses adresses IP source et de destination d'origine lorsqu'il traverse cette connexion.

5. Réception réseau sur site

Le paquet arrive à votre routeur local sur site. Le routeur consulte sa table de routage pour déterminer comment atteindre l'adresse 10.85.1.23. Pour que cela fonctionne, votre réseau local doit disposer de routes pour le pod CIDRs qui dirigent le trafic vers le nœud hybride approprié.

Dans ce cas, la table de routage du routeur contient une entrée indiquant que le `10.85.1.0/24` sous-réseau est accessible via le nœud hybride avec IP `10.80.0.2` :

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

6. Livraison au nœud

Sur la base de l'entrée de la table de routage, le routeur transmet le paquet au nœud hybride (`10.80.0.2`). Lorsque le paquet arrive au nœud, son apparence est la même que lorsque le serveur d'API EKS Kubernetes l'a envoyé, l'adresse IP de destination étant toujours l'adresse IP du pod.

7. Traitement CNI

La pile réseau du nœud reçoit le paquet et, voyant que l'adresse IP de destination n'est pas la propre adresse IP du nœud, le transmet au CNI pour traitement. Le CNI identifie que l'adresse IP de destination appartient à un pod exécuté localement sur ce nœud et transmet le paquet au pod approprié via les interfaces virtuelles appropriées :

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

Le serveur Webhook du pod reçoit la demande et la traite.

Réponse

Une fois que le module webhook a traité la demande, il renvoie une réponse en suivant le même chemin en sens inverse :

7. Pod envoie une réponse

Le module webhook crée un paquet de réponse avec sa propre adresse IP comme source et le demandeur d'origine (le plan de contrôle EKS ENI) comme destination :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.23 | Src: 8443           |                 |
| Dst: 10.0.0.132 | Dst: 41892          |                 |
+-----------------+---------------------+-----------------+
```

Le CNI identifie que ce paquet est destiné à un réseau externe (et non à un pod local) et le transmet à la pile réseau du nœud en préservant l'adresse IP source d'origine.

6. Traitement du réseau de nœuds

Le nœud détermine que l'adresse IP de destination (`10.0.0.132`) ne se trouve pas dans le réseau local et transmet le paquet à sa passerelle par défaut (le routeur local).

5. Routage du routeur local

Le routeur local consulte sa table de routage et détermine que l'adresse IP de destination (`10.0.0.132`) appartient au VPC CIDR `10.0.0.0/16` (). Il transmet le paquet à la passerelle qui se connecte à AWS.

4. Transit transfrontalier

Le paquet revient via la même connexion sur site vers le VPC, franchissant la limite du cloud dans le sens inverse.

3. Routage VPC

Lorsque le paquet arrive dans le VPC, les tables de routage indiquent que l'adresse IP de destination appartient à un sous-réseau du VPC. Le paquet est routé en conséquence au sein du VPC.

2. et 1. Plan de commande EKS ENI Reception

Le paquet atteint l'ENI rattaché au serveur API EKS Kubernetes, complétant ainsi l'aller-retour. Le serveur d'API reçoit la réponse du webhook et continue de traiter la demande d'API d'origine en fonction de cette réponse.

Ce flux de trafic montre pourquoi le module distant CIDRs doit être correctement configuré et routé :

- Le VPC doit disposer de routes pour le pod distant CIDRs pointant vers la passerelle locale
- Votre réseau sur site doit disposer de routes pour les pods CIDRs qui dirigent le trafic vers les nœuds spécifiques hébergeant ces pods
- Sans cette configuration de routage, les webhooks et autres services similaires exécutés dans des pods sur des nœuds hybrides ne seraient pas accessibles depuis le plan de contrôle EKS.

Pod-to-Pod exécution sur des nœuds hybrides

Cette section explique comment les pods exécutés sur différents nœuds hybrides communiquent entre eux. Cet exemple suppose que votre CNI utilise VXLAN pour l'encapsulation, ce qui est

courant pour Cilium CNIs ou Calico. Le processus global est similaire pour d'autres protocoles d'encapsulation tels que Geneve ou. IP-in-IP

Demande

1. Le module A initie la communication

Le Pod A (`10.85.1.56`) sur le Nœud 1 souhaite envoyer du trafic vers le Pod B (`10.85.2.67`) sur le Nœud 2. Le paquet initial ressemble à ceci :

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

2. CNI intercepte et traite le paquet

Lorsque le paquet du Pod A quitte son espace de noms réseau, le CNI l'intercepte. Le CNI consulte sa table de routage et détermine : - L'adresse IP de destination (`10.85.2.67`) appartient au CIDR du pod - Cette adresse IP ne se trouve pas sur le nœud local mais appartient au nœud 2 (`10.80.0.3`) - Le paquet doit être encapsulé avec du VXLAN.

La décision d'encapsuler est essentielle car le réseau physique sous-jacent ne sait pas comment acheminer CIDRs directement le pod. Il sait uniquement comment acheminer le trafic entre les nœuds IPs.

Le CNI encapsule l'intégralité du paquet d'origine dans une trame VXLAN. Cela crée effectivement un « paquet dans un paquet » avec de nouveaux en-têtes :

```
+-----------------+----------------+--------------+------------
+---------------------------+
| Outer Ethernet  | Outer IP       | Outer UDP    | VXLAN      | Original Pod-to-Pod
    |
| Src: Node1 MAC  | Src: 10.80.0.2 | Src: Random  | VNI: 42    | Packet (unchanged
    |
| Dst: Router MAC | Dst: 10.80.0.3 | Dst: 8472    |            | from above)
    |
+-----------------+----------------+--------------+------------
+---------------------------+
```

Points clés de cette encapsulation : - Le paquet externe est adressé du nœud 1 (`10.80.0.2`) au nœud 2 (`10.80.0.3`) - Le port UDP `8472` est le port VXLAN utilisé par défaut par Cilium - L'identifiant réseau VXLAN (VNI) identifie le réseau de superposition auquel appartient ce paquet - L'intégralité du paquet d'origine (avec l'adresse IP du pod A comme source et l'adresse IP du pod B comme destination) est préservée intacte à l'intérieur

Le paquet encapsulé entre maintenant dans la pile réseau normale du nœud 1 et est traité de la même manière que tout autre paquet :

1. Traitement réseau des nœuds : la pile réseau du nœud 1 achemine le paquet en fonction de sa destination (`10.80.0.3`)
2. Livraison sur le réseau local :
   - Si les deux nœuds sont sur le même réseau de couche 2, le paquet est envoyé directement au nœud 2
   - S'ils se trouvent sur des sous-réseaux différents, le paquet est d'abord transféré au routeur local
3. Gestion du routeur : le routeur transmet le paquet en fonction de sa table de routage et le livre au nœud 2

3. Traitement des nœuds de réception

Lorsque le paquet encapsulé arrive au nœud 2 (`10.80.0.3`) :

1. La pile réseau du nœud le reçoit et l'identifie comme un paquet VXLAN (port UDP) `4789`
2. Le paquet est transmis à l'interface VXLAN du CNI pour traitement

4. Décapsulation VXLAN

Le CNI du nœud 2 traite le paquet VXLAN :

1. Il supprime les en-têtes extérieurs (Ethernet, IP, UDP et VXLAN)
2. Il extrait le paquet intérieur d'origine
3. Le paquet a maintenant retrouvé sa forme d'origine :

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
```

```
+-----------------+----------------+------------+----------------+
```

Le CNI du nœud 2 examine l'adresse IP de destination (`10.85.2.67`) et :

1. Identifie que cette adresse IP appartient à un pod local
2. Achemine le paquet via les interfaces virtuelles appropriées
3. Fournit le paquet à l'espace de noms réseau du Pod B

Réponse

Lorsque le Pod B répond au Pod A, l'ensemble du processus se déroule en sens inverse :

4. Le Pod B envoie un paquet au Pod A (`10.85.1.56`)
5. Le CNI du nœud 2 l'encapsule avec VXLAN, en définissant la destination sur le nœud 1 () `10.80.0.2`
6. Le paquet encapsulé est livré au nœud 1
7. Le CNI du nœud 1 le décapsule et fournit la réponse initiale au module A

Des pods sur des nœuds cloud vers des pods sur des nœuds hybrides (trafic est-ouest)

Demande

1. Le module A initie la communication

Le pod A (`10.0.0.56`) sur le EC2 nœud souhaite envoyer du trafic vers le pod B (`10.85.1.56`) sur le nœud hybride. Le paquet initial ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.56  | Src: 52390 (random) |                 |
| Dst: 10.85.1.56 | Dst: 8080           |                 |
+-----------------+---------------------+-----------------+
```

Avec le VPC CNI, le Pod A possède une adresse IP provenant du VPC CIDR et est directement attaché à un ENI sur l'instance. EC2 L'espace de noms réseau du pod est connecté au réseau VPC, de sorte que le paquet entre directement dans l'infrastructure de routage VPC.

2. Routage VPC

La table de routage VPC contient une route spécifique pour le Remote Pod CIDR (`10.85.0.0/16`), dirigeant ce trafic vers la passerelle : VPC-to-onprem

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

Sur la base de cette règle de routage, le paquet est dirigé vers la passerelle qui se connecte à votre réseau local.

3. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau local. Le paquet conserve ses adresses IP source et de destination d'origine tout au long de ce transit.

4. Réception réseau sur site

Le paquet arrive à votre routeur local sur site. Le routeur consulte sa table de routage pour déterminer le saut suivant pour atteindre l'adresse 10.85.1.56. Votre routeur local doit disposer de routes pour le pod CIDRs qui dirigent le trafic vers le nœud hybride approprié.

La table du routeur contient une entrée indiquant que le `10.85.1.0/24` sous-réseau est accessible via le nœud hybride avec IP `10.80.0.2` :

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

5. Traitement du réseau de nœuds

Le routeur transmet le paquet au nœud hybride (`10.80.0.2`). Lorsque le paquet arrive au nœud, il a toujours l'adresse IP du Pod A comme source et l'IP du Pod B comme destination.

6. Traitement CNI

La pile réseau du nœud reçoit le paquet et, voyant que l'adresse IP de destination n'est pas la sienne, le transmet au CNI pour traitement. Le CNI identifie que l'adresse IP de destination appartient à un pod exécuté localement sur ce nœud et transmet le paquet au pod approprié via les interfaces virtuelles appropriées :

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

Le module B reçoit le paquet et le traite selon les besoins.

Réponse

6. Le module B envoie une réponse

Le module B crée un paquet de réponse avec sa propre adresse IP comme source et l'adresse IP du module A comme destination :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 8080           |                 |
| Dst: 10.0.0.56  | Dst: 52390          |                 |
+-----------------+---------------------+-----------------+
```

Le CNI identifie que ce paquet est destiné à un réseau externe et le transmet à la pile réseau du nœud.

5. Traitement du réseau de nœuds

Le nœud détermine que l'adresse IP de destination (`10.0.0.56`) n'appartient pas au réseau local et transmet le paquet à sa passerelle par défaut (le routeur local).

4. Routage du routeur local

Le routeur local consulte sa table de routage et détermine que l'adresse IP de destination (`10.0.0.56`) appartient au VPC CIDR `10.0.0.0/16` (). Il transmet le paquet à la passerelle qui se connecte à AWS.

3. Transit transfrontalier

Le paquet revient via la même connexion sur site vers le VPC, franchissant la limite du cloud dans le sens inverse.

2. Routage VPC

Lorsque le paquet arrive dans le VPC, le système de routage identifie que l'adresse IP de destination appartient à un sous-réseau du VPC. Le paquet est acheminé via le réseau VPC vers l'instance hébergeant EC2 le Pod A.

1. Le module A reçoit une réponse

Le paquet arrive à l' EC2 instance et est livré directement au Pod A via son ENI attaché. Comme le CNI du VPC n'utilise pas de réseau superposé pour les pods du VPC, aucune décapsulation supplémentaire n'est nécessaire : le paquet arrive avec ses en-têtes d'origine intacts.

Ce flux de trafic est-ouest montre pourquoi le module distant CIDRs doit être correctement configuré et routable dans les deux sens :

- Le VPC doit disposer de routes pour le pod distant CIDRs pointant vers la passerelle locale
- Votre réseau local doit disposer de routes pour les pods CIDRs qui dirigent le trafic vers les nœuds spécifiques hébergeant ces pods.

`nodeadm`Référence des nœuds hybrides

La CLI Amazon EKS Hybrid Nodes (`nodeadm`) simplifie l'installation, la configuration, l'enregistrement et la désinstallation des composants des nœuds hybrides. Vous pouvez inclure des images `nodeadm` dans votre système d'exploitation pour automatiser le bootstrap du nœud hybride, voir the section called "Préparer le système d'exploitation" pour plus d'informations.

La `nodeadm` version pour les nœuds hybrides est différente de la `nodeadm` version utilisée pour démarrer les EC2 instances Amazon en tant que nœuds dans des clusters Amazon EKS. Suivez la

documentation et les références pour la `nodeadm` version appropriée. Cette page de documentation concerne la `nodeadm` version des nœuds hybrides.

Le code source des nœuds hybrides `nodeadm` est publié dans le référentiel https://github.com/aws/eks-hybridGitHub .

> **Important**
>
> Vous devez exécuter `nodeadm` avec un utilisateur qui possède root/sudo des privilèges.

Télécharger **`nodeadm`**

La version des nœuds hybrides de `nodeadm` est hébergée sur Amazon S3, géré par Amazon CloudFront. Pour effectuer l'installation `nodeadm` sur chaque hôte local, vous pouvez exécuter la commande suivante à partir de vos hôtes locaux.

Pour les hôtes x86_64

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

Pour les hôtes ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

Ajoutez une autorisation de fichier exécutable au fichier binaire téléchargé sur chaque hôte.

```
chmod +x nodeadm
```

`nodeadm install`

La `nodeadm install` commande est utilisée pour installer les artefacts et les dépendances nécessaires pour exécuter et joindre des nœuds hybrides à un cluster Amazon EKS. La `nodeadm install` commande peut être exécutée individuellement sur chaque nœud hybride ou peut être exécutée pendant les pipelines de génération d'images pour préinstaller les dépendances des nœuds hybrides dans les images du système d'exploitation.

Utilisation

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

Arguments positionnels

(Obligatoire) `KUBERNETES_VERSION` La version majeure et mineure d'EKS Kubernetes à installer, par exemple `1.32`

Indicateurs

Nom	Obligatoire	Description
`-p`, `--credential-provider`	TRUE	Fournisseur d'informations d'identification à installer. Les valeurs prises en charge sont `iam-ra` et `ssm`. Pour plus d'informations, consultez the section called "Préparer les informations d'identification".
`-s`, `--containerd-source`	FALSE	Source pour `containerd`. `nodeadm` prend en charge l'installation `containerd` depuis la distribution du système d'exploitation, les packages Docker et le fait d'ignorer l'`containerd` installation. Valeurs `distro`- Il s'agit de la valeur par défaut. `nodeadm` installera le `containerd` package distribué par le système d'exploitation du nœud. `distro` n'est pas une valeur prise en charge pour les

Nom	Obligatoire	Description
		systèmes d'exploitation Red Hat Enterprise Linux (RHEL). `docker`- `nodeadm` installer a le `containerd` package créé et distribué par Docker. `dockern`'est pas une valeur prise en charge pour Amazon Linux 2023 `none`- n'`nodeadm`installer a pas le `containerd` package. Vous devez procéder à l'installation manuelle `containerd` avant de l'exécuter`nodeadm init`.
`-r`, `--region`	FALSE	Spécifie la AWS région pour le téléchargement d'artefacts tels que l'agent SSM. La valeur par défaut est `us-west-2` .
`-t`, `--timeout`	FALSE	Durée maximale de la commande d'installation. L'entrée suit le format de durée. Par exemple `1h23m`. Le délai de téléchargement par défaut pour la commande d'installation est défini sur 20 minutes.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres d'indicat eur, de sous-commande et de valeur de position disponibles.

Exemples

Installez la version Kubernetes avec AWS Systems `1.32` Manager (SSM) comme fournisseur d'informations d'identification

```
nodeadm install 1.32 --credential-provider ssm
```

Installez la version Kubernetes avec AWS Systems `1.32` Manager (SSM) comme fournisseur d'informations d'identification, Docker comme source containerd, avec un délai de téléchargement de 20 minutes.

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

Installez la version Kubernetes `1.32` avec AWS IAM Roles Anywhere comme fournisseur d'informations d'identification

```
nodeadm install 1.32 --credential-provider iam-ra
```

`nodeadm config check`

La `nodeadm config check` commande vérifie la présence d'erreurs dans la configuration du nœud fournie. Cette commande peut être utilisée pour vérifier et valider l'exactitude d'un fichier de configuration de nœud hybride.

Utilisation

```
nodeadm config check [flags]
```

Indicateurs

Nom	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de la configuration de nodeadm. Pour les nœuds hybrides, l'entrée doit suivre une URI avec schéma de fichier.

Nom	Obligatoire	Description
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres d'indicateur, de sous-commande et de valeur de position disponibles.

Exemples

```
nodeadm config check -c file://nodeConfig.yaml
```

`nodeadm init`

La `nodeadm init` commande démarre et connecte le nœud hybride au cluster Amazon EKS configuré. Consultez the section called "Config de nœud pour les activations hybrides SSM" ou the section called "Config de nœud pour les rôles IAM n'importe où" pour plus de détails sur la façon de configurer le `nodeConfig.yaml` fichier.

Utilisation

```
nodeadm init [flags]
```

Indicateurs

Nom	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de `nodeadm` configuration. Pour les nœuds hybrides, l'entrée doit suivre une URI avec schéma de fichier.
`-s`, `--skip`	FALSE	Phases de `init` à ignorer. Il n'est pas recommandé de sauter l'une ou l'autre des phases, sauf si cela permet de résoudre un problème. Valeurs

Nom	Obligatoire	Description
		`install-validation` ignore si la commande d'installation précédente a été exécutée correctement. `cni-validation` ignore si les ports VXLAN de Cilium ou Calico CNI sont ouverts si le pare-feu est activé sur le nœud `node-ip-validation` ignore si l'adresse IP du nœud correspond à un CIDR dans les réseaux de nœuds distants
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres d'indicateur, de sous-commande et de valeur de position disponibles.

Exemples

```
nodeadm init -c file://nodeConfig.yaml
```

`nodeadm upgrade`

La `nodeadm upgrade` commande met à niveau tous les artefacts installés vers la dernière version et démarre le nœud pour configurer les artefacts mis à niveau et rejoindre le cluster EKS sur AWS. La mise à niveau est une commande qui perturbe les charges de travail exécutées sur le nœud. Déplacez vos charges de travail vers un autre nœud avant d'exécuter la mise à niveau.

Utilisation

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

Arguments positionnels

(Obligatoire) `KUBERNETES_VERSION` La version majeure et mineure d'EKS Kubernetes à installer, par exemple `1.32`

Indicateurs

Nom	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de `nodeadm` configuration. Pour les nœuds hybrides, l'entrée doit suivre une URI avec schéma de fichier.
`-t`, `--timeout`	FALSE	Délai de téléchargement des artefacts. L'entrée suit le format de durée. Par exemple 1h23. Le délai de téléchargement par défaut pour la commande de mise à niveau est défini sur 10 minutes.
`-s`, `--skip`	FALSE	Les phases de mise à niveau doivent être ignorées. Il n'est pas recommandé de sauter une phase, sauf si cela permet de résoudre un problème. Valeurs `pod-validation` ignore si tous les pods non exécutés sur le nœud, à l'exception des ensembles de démons et des pods statiques. `node-validation` ignore la vérification si le nœud a été bouclé.

Nom	Obligatoire	Description
		`init-validation` ignore si le nœud a été initialisé avec succès avant d'exécuter la mise à niveau.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres d'indicat eur, de sous-commande et de valeur de position disponibles.

Exemples

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

`nodeadm uninstall`

La `nodeadm uninstall` commande arrête et supprime les artefacts `nodeadm` installés pendant l'installation`nodeadm install`, y compris le kubelet et le containerd. Notez que la commande de désinstallation ne vide ni ne supprime vos nœuds hybrides de votre cluster. Vous devez exécuter les opérations de vidange et de suppression séparément, voir the section called "Supprimer des nœuds hybrides" pour plus d'informations. Par défaut, ne `nodeadm uninstall` se poursuivra pas s'il reste des pods sur le nœud. De même, `nodeadm uninstall` ne supprime pas les dépendances CNI ni les dépendances des autres modules complémentaires Kubernetes que vous exécutez sur votre cluster. Pour supprimer complètement l'installation CNI de votre hôte, consultez les instructions à the section called "Configurer CNI" l'adresse. Si vous utilisez des activations hybrides AWS SSM comme fournisseur d'informations d'identification sur site, la `nodeadm uninstall` commande annule l'enregistrement de vos hôtes en tant qu'instances gérées par SSM. AWS

Utilisation

```
nodeadm uninstall [flags]
```

Indicateurs

Nom	Obligatoire	Description
`-s`, `--skip`	FALSE	Les phases de désinstal lation doivent être ignorées. Il n'est pas recommandé de sauter l'une ou l'autre des phases, sauf si cela permet de résoudre un problème. Valeurs `pod-validation` ignore si tous les pods non exécutés sur le nœud, à l'exception des ensembles de démons et des pods statiques. `node-validation` ignore la vérification si le nœud a été bouclé. `init-validation` ignore si le nœud a été initialisé avec succès avant d'exécuter la désinstallation.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres d'indicat eur, de sous-commande et de valeur de position disponibles.
`-f`, `--force`	FALSE	Forcer la suppression de répertoires supplémentaires susceptibles de contenir des fichiers restants provenant de Kubernetes et de composants CNI.

Nom	Obligatoire	Description
		AVERTISSEMENT Cela supprimera tout le contenu des répertoires Kubernetes et CNI par défaut (`/var/lib/cni` ,, `/etc/cni/net.d` etc.). N'utilisez pas cet indicateur si vous stockez vos propres données dans ces emplacements. À partir de nodeadm`v1.0.9`, la `./nodeadm uninstall --skip node-validation,pod-validation --force` commande ne supprime plus le répertoire. `/var/lib/kubelet` Cela est dû au fait qu'il peut contenir des volumes Pod et des répertoires volume-subpath qui incluent parfois le système de fichiers du nœud monté. Conseils de manipulation en toute sécurité - La suppression des chemins montés peut entraîner la suppression accidentelle du système de fichiers du nœud monté réel. Avant de supprimer manuellement le `/var/lib/kubelet` répertoire, inspectez soigneusement

Nom	Obligatoire	Description
		tous les montages actifs et démontez les volumes en toute sécurité pour éviter toute perte de données.

Exemples

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

`nodeadm debug`

La `nodeadm debug` commande peut être utilisée pour résoudre les problèmes liés aux nœuds hybrides défectueux ou mal configurés. Il confirme que les exigences suivantes sont en place.

- Le nœud dispose d'un accès réseau aux informations requises AWS APIs pour obtenir les informations d'identification,
- Le nœud est en mesure d'obtenir des AWS informations d'identification pour le rôle IAM des nœuds hybrides configuré,
- Le nœud dispose d'un accès réseau au point de terminaison de l'API EKS Kubernetes et de la validité du certificat du point de terminaison de l'API EKS Kubernetes,
- Le nœud est capable de s'authentifier auprès du cluster EKS, son identité dans le cluster est valide et le nœud a accès au cluster EKS via le VPC configuré pour le cluster EKS.

Si des erreurs sont détectées, le résultat de la commande suggère des étapes de dépannage. Certaines étapes de validation montrent des processus secondaires. En cas d'échec, le résultat est affiché dans une section stderr sous l'erreur de validation.

Utilisation

```
nodeadm debug [flags]
```

Indicateurs

Nom	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de `nodeadm` configuration. Pour les nœuds hybrides, l'entrée doit suivre une URI avec schéma de fichier.
`--no-color`	FALSE	Désactive la sortie couleur. Utile pour l'automatisation.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres d'indicateur, de sous-commande et de valeur de position disponibles.

Exemples

```
nodeadm debug -c file://nodeConfig.yaml
```

Emplacements des fichiers Nodeadm

installation de nodeadm

Lors de l'exécution`nodeadm install`, les fichiers et emplacements de fichiers suivants sont configurés.

Artefact	Chemin
CLI IAM Roles Anywhere	/usr/local/bin/aws_assistant à la signature
binaire Kubelet	/usr/bin/kubelet
binaire Kubectl	usr/local/bin/kubectl
Fournisseur d'informations d'identification ECR	/etc/eks/image-credential-provider/ecr-fournisseur d'informations d'identification
AWS Authentificateur IAM	/usr/local/bin/aws-iam-authentificateur

Artefact	Chemin
CLI de configuration SSM	/opt/ssm/ssm-setup-cli
SSM Agent	Sur Ubuntu -/snap/amazon-ssm-agent/current/amazon-ssm-agent Sur RHEL et AL2 023 -/-ssm-agent usr/bin/amazon
Conteneurs	Sur Ubuntu et AL2 023 -/usr/bin/containerd Sur RHEL - /bin/containers
Iptables	Sur Ubuntu et AL2 023 -/usr/sbin/iptables Sur RHEL - /sbin/iptables
Plug-ins CNI	/opt/cni/bin
suivi des artefacts installés	/opt/nodeadm/tracker

initialisation nodeadm

Lors de l'exécution`nodeadm init`, les fichiers et emplacements de fichiers suivants sont configurés.

Nom	Chemin
Kubelet kubeconfig	/var/lib/kubelet/kubeconfig
Configuration de Kubelet	/etc/kubernetes/kubelet/config.json
unité Kubelet Systemd	/etc/systemd/system/kubelet.service
Configuration du fournisseur d'informations d'identification d'image	/etc/eks/image-credential-provider/config.json
fichier d'environnement Kubelet	/etc/eks/kubelet/environment
Certificats Kubelet	/etc/kubernetes/pki/ca.crt

Nom	Chemin
Configuration du conteneur	/etc/containerd/config.toml
Configuration des modules du noyau conteneur isé	/etc/modules-load.d/contianerd.conf
AWS fichier de configuration	/etc/aws/hybrid/config
AWS fichier d'informations d'identification (si le fichier d'informations d'identification est activé)	/eks-hybrid/.aws/identifiants
AWS unité système d'aide à la signature	/etc/systemd/system/aws_signing_helper_update.service
fichier de configuration Sysctl	/etc/sysctl.d/99-nodeadm.conf
Certificats CA	/etc/ssl/certs/ca-certificates.crt
Fichier clé GPG	/etc/apt/keyrings/docker.asc
Fichier source du dépôt Docker	/etc/apt/sources.list.d/docker.liste

Config de nœud pour les activations hybrides SSM

Voici un exemple `nodeConfig.yaml` d'utilisation des activations hybrides AWS SSM pour les informations d'identification des nœuds hybrides.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Config de nœud pour les rôles IAM n'importe où

Voici un `nodeConfig.yaml` exemple d'identifiant AWS IAM Roles Anywhere pour les nœuds hybrides.

Lorsque vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification sur site, celles que `nodeName` vous utilisez dans votre `nodeadm` configuration doivent correspondre aux autorisations que vous avez définies pour votre rôle IAM Hybrid Nodes. Par exemple, si vos autorisations pour le rôle IAM de Hybrid Nodes autorisent uniquement AWS IAM Roles Anywhere à assumer le rôle lorsque le nom de session du rôle est égal au CN du certificat hôte, le `nodeName` CN de votre `nodeadm` configuration doit être le même que le CN de vos certificats. Le `nodeName` texte que vous utilisez ne peut pas comporter plus de 64 caractères. Pour de plus amples informations, veuillez consulter the section called “Préparer les informations d'identification”.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:              # Name of the EKS cluster
    region:            # AWS Region where the EKS cluster resides
  hybrid:
    iamRolesAnywhere:
      nodeName:        # Name of the node
      trustAnchorArn:  # ARN of the IAM Roles Anywhere trust anchor
      profileArn:      # ARN of the IAM Roles Anywhere profile
      roleArn:         # ARN of the Hybrid Nodes IAM role
      certificatePath: # Path to the certificate file to authenticate with the IAM
 Roles Anywhere trust anchor
      privateKeyPath:  # Path to the private key file for the certificate
```

Config de nœud pour personnaliser kubelet (facultatif)

Vous pouvez transmettre la configuration et les drapeaux de Kubelet dans votre `nodeadm` configuration. Consultez l'exemple ci-dessous pour savoir comment ajouter une étiquette de nœud `abc.amazonaws.com/test-label` et une configuration supplémentaires pour un réglage `shutdownGracePeriod` sur 30 secondes.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
```

```
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  kubelet:
    config:           # Map of kubelet config and values
       shutdownGracePeriod: 30s
    flags:            # List of kubelet flags
       - --node-labels=abc.company.com/test-label=true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Config de nœud pour personnaliser les containerd (facultatif)

Vous pouvez intégrer une configuration de containerd personnalisée dans votre `nodeadm` configuration. La configuration containerd pour `nodeadm` accepte le TOML en ligne. Consultez l'exemple ci-dessous pour savoir comment configurer containerd pour désactiver la suppression des couches d'images décompressées dans le magasin de contenu containerd.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri".containerd]
       discard_unpacked_layers = false
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Vous pouvez également utiliser la configuration containerd pour activer SELinux le support. SELinux Lorsque cette option est activée sur containerd, assurez-vous que les pods planifiés sur le nœud disposent du SecurityContext approprié et sont activés. seLinuxOptions Vous trouverez plus d'informations sur la configuration d'un contexte de sécurité dans la documentation de Kubernetes.

> **Note**
>
> Red Hat Enterprise Linux (RHEL) 8 et RHEL 9 sont SELinux activés par défaut et définis sur strict sur l'hôte. Amazon Linux 2023 est SELinux activé par défaut et est réglé sur le mode permissif. Lorsqu'il SELinux est défini sur le mode permissif sur l'hôte, son activation sur containerd ne bloquera pas les demandes mais les enregistrera conformément à la SELinux configuration de l'hôte.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri"]
       enable_selinux = true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Résolution des problèmes liés aux nœuds hybrides

Cette rubrique décrit certaines erreurs courantes que vous pouvez rencontrer lors de l'utilisation des nœuds hybrides Amazon EKS et explique comment les corriger. Pour d'autres informations de dépannage, consultez le Résolution des problèmes tag Knowledge Center pour Amazon EKS sur AWS Re:Post. Si vous ne parvenez pas à résoudre le problème, contactez AWS le Support.

Résolution des problèmes liés aux nœuds avec **`nodeadm debug`** Vous pouvez exécuter la `nodeadm debug` commande depuis vos nœuds hybrides pour vérifier que les exigences relatives au réseau et aux informations d'identification sont satisfaites. Pour plus d'informations sur la `nodeadm debug` commande, consultezthe section called "Nœuds hybrides nodeadm".

Détectez les problèmes liés à vos nœuds hybrides grâce à des informations sur les clusters Amazon EKS Cluster Insights inclut des contrôles d'informations qui détectent les problèmes courants liés à la configuration des nœuds hybrides EKS dans votre cluster. Vous pouvez consulter les résultats de toutes les vérifications d'aperçu à partir de la AWS Management Console AWS CLI et du AWS

SDKs. Pour plus d'informations sur les informations sur les clusters, consultez the section called "Informations sur le cluster".

Dépannage de l'installation de nœuds hybrides

Les rubriques de dépannage suivantes concernent l'installation des dépendances des nœuds hybrides sur les hôtes à l'aide de la `nodeadm install` commande.

`nodeadm` échec de la commande **`must run as root`**

La `nodeadm install` commande doit être exécutée avec un utilisateur disposant de droits root ou de `sudo` privilèges sur votre hôte. Si vous exécutez `nodeadm install` avec un utilisateur qui n'a ni root ni `sudo` privilèges, vous verrez le message d'erreur suivant s'afficher dans le `nodeadm` résultat.

```
"msg":"Command failed","error":"must run as root"
```

Impossible de se connecter aux dépendances

La `nodeadm install` commande installe les dépendances requises pour les nœuds hybrides. Les dépendances des nœuds hybrides incluent `containerd`, `kubelet` `kubectl`, et les composants AWS SSM ou AWS IAM Roles Anywhere. Vous devez avoir accès à partir de l'endroit où vous courez `nodeadm install` pour télécharger ces dépendances. Pour plus d'informations sur la liste des emplacements auxquels vous devez être en mesure d'accéder, consultez the section called "Préparer la mise en réseau". Si vous n'y avez pas accès, vous verrez des erreurs similaires aux suivantes dans le `nodeadm install` résultat.

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved
 for http request"
```

Impossible de mettre à jour le gestionnaire de packages

La `nodeadm install` commande s'exécute `apt update` `yum update` ou `dnf update` avant l'installation des dépendances des nœuds hybrides. Si cette étape échoue, des erreurs similaires aux suivantes peuvent s'afficher. Pour remédier à cette situation, vous pouvez exécuter `apt update` `yum update` ou `dnf update` avant l'exécution `nodeadm install` ou essayer de le `nodeadm install` réexécuter.

```
failed to run update using package manager
```

Délai d'expiration ou délai de mise en contexte dépassé

Lors de l'exécution`nodeadm install`, si vous rencontrez des problèmes à différentes étapes du processus d'installation avec des erreurs indiquant un délai d'expiration ou un délai contextuel dépassé, il se peut que votre connexion soit lente et empêche l'installation des dépendances des nœuds hybrides avant que les délais ne soient atteints. Pour contourner ces problèmes, vous pouvez essayer d'utiliser l'`--timeout`indicateur `nodeadm` pour prolonger la durée des délais de téléchargement des dépendances.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --timeout 20m0s
```

Dépannage des problèmes liés à la connexion

Les rubriques de résolution des problèmes de cette section concernent le processus de connexion des nœuds hybrides aux clusters EKS à l'aide de la `nodeadm init` commande.

Erreurs de fonctionnement ou schéma non pris en charge

Lors de l'exécution`nodeadm init`, si vous constatez des erreurs liées `nodeConfig.yaml` à `operation error` ou`unsupported scheme`, vérifiez que le format et la transmission sont corrects. `nodeadm` Pour plus d'informations sur le format et les options pour`nodeConfig.yaml`, voirthe section called “Nœuds hybrides nodeadm”.

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled,
 context deadline exceeded"
```

Nœuds hybrides (rôle IAM) : autorisations manquantes pour l'action **`eks:DescribeCluster`**

Lors de l'exécution`nodeadm init`, `nodeadm` tente de recueillir des informations sur votre cluster EKS en appelant Describe Cluster. Si votre rôle Hybrid Nodes IAM n'est pas autorisé à effectuer cette `eks:DescribeCluster` action. Pour plus d'informations sur les autorisations requises pour le rôle IAM des nœuds hybrides, consultezthe section called “Préparer les informations d'identification”.

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response
 error StatusCode: 403 ... AccessDeniedException"
```

L'adresse IP du nœud ne figure pas dans le réseau de nœuds distants CIDR

Lors de l'exécution`nodeadm init`, vous pouvez rencontrer une erreur si l'adresse IP du nœud ne se trouve pas dans le réseau de nœuds distants spécifié CIDRs. L'erreur ressemblera à l'exemple suivant :

```
node IP 10.18.0.1 is not in any of the remote network CIDR blocks [10.0.0.0/16
 192.168.0.0/16]
```

Cet exemple montre un nœud dont l'adresse IP est 10.18.0.1 qui tente de rejoindre un cluster doté du réseau distant CIDRs 10.0.0.0/16 et 192.168.0.0/16. L'erreur se produit car 10.18.0.1 ne se situe dans aucune des plages.

Vérifiez que vous avez correctement configuré votre adresse IP `RemoteNodeNetworks` pour inclure toutes les adresses IP des nœuds. Pour plus d'informations sur la configuration réseau, consultezthe section called "Préparer la mise en réseau".

- Exécutez la commande suivante dans la région où se trouve votre cluster pour vérifier vos `RemoteNodeNetwork` configurations. Vérifiez que les blocs CIDR répertoriés dans la sortie incluent la plage d'adresses IP de votre nœud et sont identiques aux blocs CIDR répertoriés dans le message d'erreur. S'ils ne correspondent pas, confirmez que le nom et la région du cluster `nodeConfig.yaml` correspondent à votre cluster prévu.

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
 cluster.remoteNetworkConfig.remoteNodeNetworks
```

- Vérifiez que vous travaillez avec le nœud prévu :
  - Vérifiez que vous êtes sur le bon nœud en vérifiant que son nom d'hôte et son adresse IP correspondent à ceux que vous souhaitez enregistrer auprès du cluster.
  - Vérifiez que ce nœud se trouve dans le bon réseau local (celui dont la plage d'adresses CIDR a été enregistrée `RemoteNodeNetwork` lors de la configuration du cluster).

Si l'adresse IP de votre nœud n'est toujours pas celle que vous attendiez, vérifiez les points suivants :

- Si vous utilisez IAM Roles Anywhere, `kubelet` effectue une recherche DNS sur IAM Roles Anywhere `nodeName` et utilise une adresse IP associée au nom du nœud, le cas échéant. Si vous conservez des entrées DNS pour vos nœuds, vérifiez qu'elles pointent vers l' IPs intérieur de votre réseau de nœuds distants CIDRs.
- Si votre nœud possède plusieurs interfaces réseau, vous `kubelet` pouvez sélectionner une interface avec une adresse IP en dehors du réseau CIDRs de votre nœud distant par défaut. Pour utiliser une autre interface, spécifiez son adresse IP à l'aide du `--node-ip` `kubelet` drapeau figurant dans votre`nodeConfig.yaml`. Pour de plus amples informations, veuillez consulter the

section called "Nœuds hybrides nodeadm". Vous pouvez consulter les interfaces réseau et les adresses IP de votre nœud en exécutant la commande suivante sur votre nœud :

```
ip addr show
```

Les nœuds hybrides n'apparaissent pas dans le cluster EKS

Si vous l'avez exécuté `nodeadm init` et que vous l'avez terminé, mais que vos nœuds hybrides n'apparaissent pas dans votre cluster, il se peut que des problèmes se produisent au niveau de la connexion réseau entre vos nœuds hybrides et le plan de contrôle EKS, que vous n'ayez pas configuré les autorisations de groupe de sécurité requises ou que vous n'ayez pas le mappage requis entre votre rôle IAM de nœuds hybrides et le contrôle d'accès basé sur les rôles (RBAC) de Kubernetes. Vous pouvez démarrer le processus de débogage en vérifiant l'état `kubelet` et les `kubelet` journaux à l'aide des commandes suivantes. Exécutez les commandes suivantes à partir des nœuds hybrides qui n'ont pas réussi à rejoindre votre cluster.

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Impossible de communiquer avec le cluster

Si votre nœud hybride n'a pas pu communiquer avec le plan de contrôle du cluster, des journaux similaires aux suivants peuvent s'afficher.

```
"Failed to ensure lease exists, will retry" err="Get ..."
```

```
"Unable to register node with API server" err="Post ..."
```

```
Failed to contact API server when waiting for CSINode publishing ... dial tcp <ip
 address>: i/o timeout
```

Si vous voyez ces messages, vérifiez les points suivants pour vous assurer qu'ils répondent aux exigences relatives aux nœuds hybrides détaillées dansthe section called "Préparer la mise en réseau".

- Vérifiez que le VPC transmis au cluster EKS possède des itinéraires vers votre Transit Gateway (TGW) ou votre passerelle privée virtuelle (VGW) pour votre nœud sur site et éventuellement votre pod. CIDRs
- Vérifiez que vous disposez d'un groupe de sécurité supplémentaire pour votre cluster EKS et que des règles d'entrée s'appliquent à votre nœud local CIDRs et, éventuellement, à votre pod. CIDRs
- Vérifiez que votre routeur local est configuré pour autoriser le trafic à destination et en provenance du plan de contrôle EKS.

Non autorisé

Si votre nœud hybride a pu communiquer avec le plan de contrôle EKS mais n'a pas pu s'enregistrer, des journaux similaires aux suivants peuvent s'afficher. Notez que la principale différence entre les messages du journal ci-dessous est l'`Unauthorized`erreur. Cela indique que le nœud n'a pas été en mesure d'effectuer ses tâches car il ne dispose pas des autorisations requises.

```
"Failed to ensure lease exists, will retry" err="Unauthorized"
```

```
"Unable to register node with API server" err="Unauthorized"
```

```
Failed to contact API server when waiting for CSINode publishing: Unauthorized
```

Si vous voyez ces messages, vérifiez les points suivants pour vous assurer qu'ils répondent aux exigences relatives aux nœuds hybrides détaillées dans the section called "Préparer les informations d'identification" etthe section called "Préparer l'accès au cluster".

- Vérifiez que l'identité des nœuds hybrides correspond au rôle IAM attendu de vos nœuds hybrides. Cela peut être fait en exécutant `sudo aws sts get-caller-identity` à partir de vos nœuds hybrides.
- Vérifiez que votre rôle IAM Hybrid Nodes dispose des autorisations requises.
- Vérifiez que dans votre cluster, vous disposez d'une entrée d'accès EKS pour votre rôle IAM Hybrid Nodes ou confirmez que vous disposez d'`aws-auth` ConfigMap une entrée pour votre rôle IAM Hybrid Nodes. Si vous utilisez des entrées d'accès EKS, vérifiez que votre entrée d'accès pour votre rôle IAM de Hybrid Nodes possède le type `HYBRID_LINUX` d'accès. Si vous utilisez le `aws-auth` ConfigMap, confirmez que votre entrée pour le rôle Hybrid Nodes IAM répond aux exigences et au format détaillés dansthe section called "Préparer l'accès au cluster".

Nœuds hybrides enregistrés auprès du cluster EKS mais affichant leur statut **`Not Ready`**

Si vos nœuds hybrides se sont enregistrés avec succès auprès de votre cluster EKS, mais que le statut des nœuds hybrides s'`Not Ready`affiche, la première chose à vérifier est le statut de votre interface réseau de conteneurs (CNI). Si vous n'avez pas installé de CNI, on s'attend à ce que vos nœuds hybrides aient un statut`Not Ready`. Une fois qu'un CNI est installé et fonctionne correctement, le statut `Ready` des nœuds est mis à jour. Si vous avez essayé d'installer un CNI mais qu'il ne fonctionne pas correctement, consultez cette the section called "Dépannage des nœuds hybrides CNI" page.

Les demandes de signature de certificat (CSRs) sont bloquées en attente

Après avoir connecté des nœuds hybrides à votre cluster EKS, si vous constatez que CSRs des nœuds hybrides sont en attente, cela signifie que ceux-ci ne répondent pas aux exigences d'approbation automatique. CSRs pour les nœuds hybrides sont automatiquement approuvés et émis si les CSRs nœuds hybrides ont été créés par un nœud portant une `eks.amazonaws.com/compute-type: hybrid` étiquette, et que le CSR possède les noms alternatifs de sujet suivants (SANs) : au moins un SAN DNS égal au nom du nœud et l'adresse IP SANs appartient au réseau de nœuds distants CIDRs.

Le profil hybride existe déjà

Si vous avez modifié votre `nodeadm` configuration et que vous tentez de réenregistrer le nœud avec la nouvelle configuration, un message d'erreur indiquant que le profil hybride existe déjà mais que son contenu a changé peut s'afficher. Au lieu de l'exécuter `nodeadm init` entre les modifications de configuration, exécutez `nodeadm uninstall` suivi d'un `nodeadm install` et`nodeadm init`. Cela garantit un nettoyage approprié en fonction des modifications de configuration.

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config
 but its contents do not align with the expected configuration"
```

Le nœud hybride n'a pas réussi à résoudre l'API privée

Après l'exécution`nodeadm init`, si vous voyez une erreur dans les `kubelet` journaux indiquant que vous n'avez pas réussi à contacter le serveur d'API EKS Kubernetes parce que c'est le casno `such host`, vous devrez peut-être modifier votre entrée DNS pour le point de terminaison de l'API EKS Kubernetes sur votre réseau local ou au niveau de l'hôte. Consultez la section Transfert de requêtes DNS entrantes vers votre VPC dans AWS la documentation de Route53.

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

Impossible d'afficher les nœuds hybrides dans la console EKS

Si vous avez enregistré vos nœuds hybrides mais que vous ne parvenez pas à les afficher dans la console EKS, vérifiez les autorisations du principal IAM que vous utilisez pour consulter la console. Le principal IAM que vous utilisez doit disposer d'autorisations IAM et Kubernetes minimales spécifiques pour afficher les ressources dans la console. Pour de plus amples informations, veuillez consulter the section called "Accès aux ressources du cluster".

Résolution des problèmes d'exécution de nœuds hybrides

Si vos nœuds hybrides étaient enregistrés auprès de votre cluster EKS, avaient un statut `Ready` puis sont passés au statut`Not Ready`, de nombreux problèmes peuvent avoir contribué à ce mauvais état, tels que le fait que le nœud ne dispose pas de ressources suffisantes pour le processeur, la mémoire ou l'espace disque disponible, ou que le nœud est déconnecté du plan de contrôle EKS. Vous pouvez suivre les étapes ci-dessous pour dépanner vos nœuds, et si vous ne parvenez pas à résoudre le problème, contactez le AWS Support.

Exécutez `nodeadm debug` à partir de vos nœuds hybrides pour vérifier que les exigences en matière de réseau et d'identification sont satisfaites. Pour plus d'informations sur la `nodeadm debug` commande, consultezthe section called "Nœuds hybrides nodeadm".

Obtenir le statut du nœud

```
kubectl get nodes -o wide
```

Vérifiez les conditions et les événements du nœud

```
kubectl describe node NODE_NAME
```

Obtenir le statut du pod

```
kubectl get pods -A -o wide
```

Vérifiez l'état et les événements du pod

```
kubectl describe pod POD_NAME
```

Vérifiez les journaux du pod

```
kubectl logs POD_NAME
```

Vérifiez les **`kubectl`** journaux

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Les sondes de durée de vie du pod échouent ou les webhooks ne fonctionnent pas

Si les applications, les modules complémentaires ou les webhooks exécutés sur vos nœuds hybrides ne démarrent pas correctement, il se peut que vous rencontriez des problèmes de réseau qui bloquent la communication avec les modules. Pour que le plan de contrôle EKS puisse contacter les webhooks exécutés sur des nœuds hybrides, vous devez configurer votre cluster EKS avec un réseau de modules distants et disposer de routes pour le CIDR de votre pod local dans la table de routage de votre VPC avec comme cible votre Transit Gateway (TGW), votre passerelle privée virtuelle (VPW) ou toute autre passerelle que vous utilisez pour connecter votre VPC à votre réseau local. Pour plus d'informations sur les exigences de mise en réseau pour les nœuds hybrides, consultezthe section called "Préparer la mise en réseau". Vous devez également autoriser ce trafic dans votre pare-feu local et vous assurer que votre routeur peut être correctement acheminé vers vos pods. Consultez the section called "Configurer les webhooks" pour plus d'informations sur les exigences relatives à l'exécution de webhooks sur des nœuds hybrides.

Un message de log du pod courant pour ce scénario est affiché ci-dessous, où l'adresse IP est l'adresse IP du cluster pour le service Kubernetes.

```
dial tcp <ip-address>:443: connect: no route to host
```

`kubectl logs`ou **`kubectl exec`** les commandes ne fonctionnent pas

Si `kubectl logs` `kubectl exec` les commandes expirent alors que `kubectl` les autres commandes aboutissent, le problème est probablement lié à la configuration du réseau distant. Ces commandes se connectent via le cluster au `kubelet` point de terminaison du nœud. Pour de plus amples informations, veuillez consulter the section called "Point de terminaison `kubelet`".

Vérifiez que votre nœud IPs et votre pod font IPs partie du réseau de nœuds distants et du réseau de nœuds distants CIDRs configurés pour votre cluster. Utilisez les commandes ci-dessous pour examiner les attributions d'adresses IP.

```
kubectl get nodes -o wide
```

```
kubectl get pods -A -o wide
```

IPs Comparez-les avec votre réseau distant configuré CIDRs pour garantir un routage correct. Pour les exigences de configuration réseau, voirthe section called "Préparer la mise en réseau".

Dépannage des nœuds hybrides CNI

Si vous rencontrez des problèmes lors du démarrage initial de Cilium ou Calico avec des nœuds hybrides, cela est le plus souvent dû à des problèmes de réseau entre les nœuds hybrides ou les pods CNI exécutés sur des nœuds hybrides et le plan de contrôle EKS. Assurez-vous que votre environnement répond aux exigences de la section Préparer le réseau pour les nœuds hybrides. Il est utile de décomposer le problème en plusieurs parties.

Configuration du cluster EKS

- Les RemotePodNetwork configurations RemoteNodeNetwork et sont-elles correctes ?

Configuration VPC

- Existe-t-il des itinéraires pour RemoteNodeNetwork et RemotePodNetwork dans la table de routage VPC avec la cible du Transit Gateway ou de la Virtual Private Gateway ?

Configuration du groupe de sécurité

- Existe-t-il des règles d'entrée et de sortie pour le RemoteNodeNetwork et ? RemotePodNetwork

Réseau local

- Existe-t-il des itinéraires et des accès vers et depuis le plan de contrôle EKS et vers et depuis les nœuds hybrides et les pods exécutés sur des nœuds hybrides ?

Configuration CNI

- Si vous utilisez un réseau superposé, la configuration du pool IP pour le CNI correspond-elle à celle RemotePodNetwork configurée pour le cluster EKS si vous utilisez des webhooks ?

Le nœud hybride a un statut **Ready** sans CNI installé

Si vos nœuds hybrides affichent un état`Ready`, mais que vous n'avez pas installé de CNI sur votre cluster, il est possible que d'anciens artefacts CNI soient présents sur vos nœuds hybrides. Par défaut, lorsque vous désinstallez Cilium et Calico à l'aide d'outils tels que Helm, les ressources sur disque ne sont pas supprimées de vos machines physiques ou virtuelles. En outre, les définitions de ressources personnalisées (CRDs) correspondantes CNIs peuvent toujours être présentes sur votre cluster à partir d'une ancienne installation. Pour plus d'informations, consultez les sections Supprimer le cilium et Supprimer le calico de. the section called "Configurer CNI"

Résolution des problèmes liés à Cilium

Si vous rencontrez des problèmes pour exécuter Cilium sur des nœuds hybrides, consultez les étapes de résolution des problèmes dans la documentation de Cilium. Les sections ci-dessous abordent les problèmes qui peuvent être propres au déploiement de Cilium sur des nœuds hybrides.

Cilium ne démarre pas

Si les agents Cilium qui s'exécutent sur chaque nœud hybride ne démarrent pas, vérifiez la présence d'erreurs dans les journaux des pods d'agents Cilium. L'agent Cilium doit être connecté au point de terminaison de l'API EKS Kubernetes pour démarrer. Le démarrage de l'agent Cilium échouera si cette connectivité n'est pas correctement configurée. Dans ce cas, vous verrez des messages de journal similaires aux suivants dans les journaux du module de l'agent Cilium.

```
msg="Unable to contact k8s api-server"
level=fatal msg="failed to start: Get \"https://<k8s-cluster-ip>:443/api/v1/namespaces/
kube-system\": dial tcp <k8s-cluster-ip>:443: i/o timeout"
```

L'agent Cilium s'exécute sur le réseau hôte. Votre cluster EKS doit être configuré `RemoteNodeNetwork` pour la connectivité Cilium. Vérifiez que vous disposez d'un groupe de sécurité supplémentaire pour votre cluster EKS avec une règle entrante pour vous`RemoteNodeNetwork`, que vous avez des itinéraires dans votre VPC pour vous et que `RemoteNodeNetwork` votre réseau sur site est correctement configuré pour permettre la connectivité au plan de contrôle EKS.

Si l'opérateur Cilium est en cours d'exécution et que certains de vos agents Cilium sont en cours d'exécution, mais pas tous, vérifiez que vous disposez d'un pod disponible IPs à allouer pour tous les nœuds de votre cluster. Vous configurez la taille de votre pod allouable CIDRs lorsque vous utilisez le pool de clusters IPAM `clusterPoolIPv4PodCIDRList` dans votre configuration Cilium.

La taille CIDR par nœud est configurée selon le `clusterPoolIPv4MaskSize` paramètre de votre configuration Cilium. Voir Extension du pool de clusters dans la documentation de Cilium pour plus d'informations.

Cilium BGP ne fonctionne pas

Si vous utilisez le plan de contrôle BGP de Cilium pour publier les adresses de vos pods ou de vos services sur votre réseau local, vous pouvez utiliser les commandes Cilium CLI suivantes pour vérifier si BGP annonce les itinéraires vers vos ressources. Pour connaître les étapes d'installation de la CLI Cilium, consultez la section Installer la CLI Cilium dans la documentation de Cilium.

Si BGP fonctionne correctement, vous devez indiquer l'état de session de vos nœuds hybrides `established` dans la sortie. Vous devrez peut-être travailler avec votre équipe réseau pour identifier les valeurs correctes pour l'AS local, l'AS homologue et l'adresse homologue de votre environnement.

```
cilium bgp peers
```

```
cilium bgp routes
```

Si vous utilisez Cilium BGP pour faire IPs de la publicité pour des Services par type`LoadBalancer`, vous devez avoir la même étiquette sur votre produit `CiliumLoadBalancerIPPool` et sur le Service, qui doit être utilisée dans le sélecteur de votre. `CiliumBGPAdvertisement` Un exemple est illustré ci-dessous. Notez que si vous utilisez Cilium BGP pour promouvoir les services avec type LoadBalancer, les IPs routes BGP peuvent être perturbées lors du redémarrage de l'agent Cilium. Pour plus d'informations, consultez la section Scénarios de défaillance dans la documentation de Cilium.

Service

```
kind: Service
apiVersion: v1
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
```

```
  selector:
    app: guestbook
  type: LoadBalancer
```

CiliumLoadBalancerIPPool

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: guestbook-pool
  labels:
    app: guestbook
spec:
  blocks:
  - cidr: <CIDR to advertise>
  serviceSelector:
    matchExpressions:
      - { key: app, operator: In, values: [ guestbook ] }
```

CiliumBGPAdvertisement

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements-guestbook
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - ExternalIP
          - LoadBalancerIP
      selector:
        matchExpressions:
          - { key: app, operator: In, values: [ guestbook ] }
```

Résolution des problèmes liés à Calico

Si vous rencontrez des problèmes pour exécuter Calico sur des nœuds hybrides, consultez les étapes de résolution des problèmes décrites dans la documentation de Calico. Les sections ci-

dessous abordent les problèmes qui peuvent être propres au déploiement de Calico sur des nœuds hybrides.

Le tableau ci-dessous récapitule les composants Calico et indique s'ils s'exécutent par défaut sur le réseau de nœuds ou de pods. Si vous avez configuré Calico pour utiliser le NAT pour le trafic de pods sortant, votre réseau local doit être configuré pour acheminer le trafic vers le CIDR de votre nœud local et vos tables de routage VPC doivent être configurées avec une route pour le CIDR de votre nœud local avec votre passerelle de transit (TGW) ou votre passerelle privée virtuelle (VGW) comme cible. Si vous ne configurez pas Calico pour utiliser le NAT pour le trafic de pod sortant, votre réseau local doit être configuré pour acheminer le trafic vers le CIDR de votre pod local et vos tables de routage VPC doivent être configurées avec une route pour le CIDR de votre pod local avec votre passerelle de transit (TGW) ou votre passerelle privée virtuelle (VGW) comme cible.

Composant	Réseau
Serveur API Calico	Nœud
Contrôleurs Calico pour Kubernetes	Pod
Agent du nœud Calico	Nœud
Calicot `typha`	Nœud
Pilote de nœud Calico CSI	Pod
Opérateur Calico	Nœud

Les ressources Calico sont planifiées ou exécutées sur des nœuds cordonés

Les ressources Calico qui ne s'exécutent pas en tant que telles DaemonSet sont soumises à des tolérances flexibles par défaut qui leur permettent d'être planifiées sur des nœuds cordonés qui ne sont pas prêts à être planifiés ou à exécuter des pods. Vous pouvez renforcer les tolérances pour les ressources autres que DaemonSet Calico en modifiant l'installation de votre opérateur pour inclure les éléments suivants.

```
installation:
  ...
  controlPlaneTolerations:
  - effect: NoExecute
```

```
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
  typhaDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
```

Résolution des problèmes d'identification

Pour les activations hybrides AWS SSM et pour AWS IAM Roles Anywhere, vous pouvez vérifier que les informations d'identification du rôle Hybrid Nodes IAM sont correctement configurées sur vos nœuds hybrides en exécutant la commande suivante depuis vos nœuds hybrides. Vérifiez que le nom du nœud et le nom du rôle IAM des nœuds hybrides correspondent à vos attentes.

```
sudo aws sts get-caller-identity
```

```
{
    "UserId": "ABCDEFGHIJKLM12345678910:<node-name>",
    "Account": "<aws-account-id>",
    "Arn": "arn:aws: sts::<aws-account-id>:assumed-role/<hybrid-nodes-iam-role/<node-name>"
}
```

AWS Résolution des problèmes liés à Systems Manager (SSM)

Si vous utilisez des activations hybrides AWS SSM pour les informations d'identification de vos nœuds hybrides, tenez compte des répertoires et artefacts SSM suivants qui sont installés sur vos nœuds hybrides par. `nodeadm` Pour plus d'informations sur l'agent SSM, consultez la section Utilisation de l'agent SSM dans le Guide de l'utilisateur de AWS Systems Manager.

Description	Emplacement
Agent SSM	Ubuntu - `/snap/amazon-ssm-agent/current/amazon-ssm-agent` RHEL et AL2 023 - `/usr/bin/amazon-ssm-agent`
Journaux SSM Agent	`/var/log/amazon/ssm`
AWS informations d'identification	`/root/.aws/credentials`
CLI de configuration SSM	`/opt/ssm/ssm-setup-cli`

Redémarrage de l'agent SSM

Certains problèmes peuvent être résolus en redémarrant l'agent SSM. Vous pouvez utiliser les commandes ci-dessous pour le redémarrer.

AL2023 et autres systèmes d'exploitation

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

Vérifiez la connectivité aux points de terminaison SSM

Vérifiez que vous pouvez vous connecter aux points de terminaison SSM à partir de vos nœuds hybrides. Pour obtenir la liste des points de terminaison SSM, consultez la section Points de terminaison AWS et quotas de Systems Manager. Remplacez `us-west-2` la commande ci-dessous par la AWS région pour votre activation hybride AWS SSM.

```
ping ssm.us-west-2.amazonaws.com
```

Afficher l'état de connexion des instances SSM enregistrées

Vous pouvez vérifier l'état de connexion des instances enregistrées avec les activations hybrides SSM à l'aide de la commande AWS CLI suivante. Remplacez l'ID de machine par l'ID de machine de votre instance.

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

Incompatibilité de la somme de contrôle de la CLI de configuration SSM

Lors de l'exécution, `nodeadm install` si vous constatez un problème lié à l'`ssm-setup-cli`incompatibilité de la somme de contrôle, vous devez vérifier qu'aucune ancienne installation SSM n'existe sur votre hôte. S'il existe d'anciennes installations SSM sur votre hôte, supprimez-les et réexécutez-les `nodeadm install` pour résoudre le problème.

```
Failed to perform agent-installation/on-prem registration: error while verifying
 installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

SSM **`InvalidActivation`**

Si vous constatez une erreur lors de l'enregistrement de votre instance auprès de AWS SSM, confirmez que le `regionactivationCode`, et `activationId` dans votre nom `nodeConfig.yaml` sont corrects. La AWS région de votre cluster EKS doit correspondre à la région de votre activation hybride SSM. Si ces valeurs sont mal configurées, une erreur similaire à la suivante peut s'afficher.

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 InvalidActivation
```

SSM **`ExpiredTokenException`** : le jeton de sécurité inclus dans la demande a expiré

Si l'agent SSM n'est pas en mesure d'actualiser les informations d'identification, un`ExpiredTokenException`. Dans ce scénario, si vous parvenez à vous connecter aux points de terminaison SSM à partir de vos nœuds hybrides, vous devrez peut-être redémarrer l'agent SSM pour forcer l'actualisation des informations d'identification.

```
"msg":"Command failed","error":"operation error SSM: DescribeInstanceInformation, https
 response error StatusCode: 400, RequestID: eee03a9e-f7cc-470a-9647-73d47e4cf0be, api
 error ExpiredTokenException: The security token included in the request is expired"
```

Erreur SSM lors de l'exécution de la commande register machine

Si une erreur s'affiche lors de l'enregistrement de la machine auprès de SSM, vous devrez peut-être l'exécuter `nodeadm install` à nouveau pour vous assurer que toutes les dépendances SSM sont correctement installées.

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
 no such file or directory"
```

SSM **`ActivationExpired`**

Lors de l'exécution`nodeadm init`, si vous constatez une erreur lors de l'enregistrement de l'instance auprès de SSM en raison d'une activation expirée, vous devez créer une nouvelle activation hybride SSM, la mettre à jour `nodeConfig.yaml` avec le `activationCode` et `activationId` de votre nouvelle activation hybride SSM, puis la réexécuter. `nodeadm init`

```
"msg":"Command failed","error":"SSM activation expired. Please use a valid activation"
```

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 ActivationExpired
```

SSM n'a pas réussi à actualiser les informations d'identification mises en cache

Si vous ne parvenez pas à actualiser les informations d'identification mises en cache, le `/root/.aws/credentials` fichier a peut-être été supprimé sur votre hôte. Vérifiez d'abord votre activation hybride SSM, assurez-vous qu'elle est active et que vos nœuds hybrides sont correctement configurés pour utiliser l'activation. Vérifiez les journaux de l'agent SSM `/var/log/amazon/ssm` et réexécutez la `nodeadm init` commande une fois que vous avez résolu le problème côté SSM.

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
 identity: get credentials: failed to refresh cached credentials"
```

Nettoyez SSM

Pour supprimer l'agent SSM de votre hôte, vous pouvez exécuter les commandes suivantes.

```
dnf remove -y amazon-ssm-agent
sudo apt remove --purge amazon-ssm-agent
snap remove amazon-ssm-agent
rm -rf /var/lib/amazon/ssm/Vault/Store/RegistrationKey
```

AWS Résolution des problèmes liés à IAM Roles Anywhere

Si vous utilisez AWS IAM Roles Anywhere pour les informations d'identification de vos nœuds hybrides, tenez compte des répertoires et artefacts suivants qui sont installés sur vos nœuds hybrides par`nodeadm`. Pour plus d'informations sur la résolution des problèmes liés à IAM Roles Anywhere, consultez la section Résolution des problèmes d'identité et d'accès à AWS IAM Roles Anywhere dans le guide de l'utilisateur d' AWS IAM Roles Anywhere.

Description	Emplacement
CLI IAM Roles Anywhere	`/usr/local/bin/aws_signing_helper`
Emplacement et nom du certificat par défaut	`/etc/iam/pki/server.pem`
Emplacement et nom de la clé par défaut	`/etc/iam/pki/server.key`

IAM Roles Anywhere n'a pas pu actualiser les informations d'identification mises en cache

Si vous ne parvenez pas à actualiser les informations d'identification mises en cache, passez en revue le contenu d'IAM Roles Anywhere `/etc/aws/hybrid/config` et confirmez qu'il a été correctement configuré dans votre `nodeadm` configuration. Confirmez que cela `/etc/iam/pki` existe. Chaque nœud doit disposer d'un certificat et d'une clé uniques. Par défaut, lorsque vous utilisez IAM Roles Anywhere comme fournisseur d'informations d'identification, utilisez cette option `nodeadm` `/etc/iam/pki/server.pem` pour l'emplacement et le nom du certificat, ainsi que `/etc/iam/pki/server.key` pour la clé privée. Vous devrez peut-être créer les répertoires avant de

placer les certificats et les clés dans les répertoires avec`sudo mkdir -p /etc/iam/pki`. Vous pouvez vérifier le contenu de votre certificat à l'aide de la commande ci-dessous.

```
openssl x509 -text -noout -in server.pem
```

```
open /etc/iam/pki/server.pem: no such file or directory
could not parse PEM data
Command failed {"error": "... get identity: get credentials: failed to refresh cached
 credentials, process provider error: error in credential_process: exit status 1"}
```

IAM Roles Anywhere n'est pas autorisé à exécuter **`sts:AssumeRole`**

Dans les `kubelet` journaux, si vous constatez un problème de refus d'accès lié à l'`sts:AssumeRole`opération lorsque vous utilisez IAM Roles Anywhere, vérifiez la politique de confiance de votre rôle IAM Hybrid Nodes pour confirmer que le principal du service IAM Roles Anywhere est autorisé à assumer le rôle IAM Hybrid Nodes. Vérifiez également que l'ARN d'ancrage de confiance est correctement configuré dans la politique de confiance de votre rôle IAM Hybrid Nodes et que votre rôle IAM Hybrid Nodes est ajouté à votre profil IAM Roles Anywhere.

```
could not get token: AccessDenied: User: ... is not authorized to perform:
 sts:AssumeRole on resource: ...
```

IAM Roles Anywhere n'est pas autorisé à définir **`roleSessionName`**

Dans les `kubelet` journaux, si vous constatez un problème de refus d'accès lié à la définition de`roleSessionName`, confirmez que vous avez défini la valeur true `acceptRoleSessionName` pour votre profil IAM Roles Anywhere.

```
AccessDeniedException: Not authorized to set roleSessionName
```

Dépannage du système d'exploitation

RHEL

Échec de l'enregistrement du gestionnaire d'autorisations ou d'abonnements

Si vous exécutez `nodeadm install` et que vous ne parvenez pas à installer les dépendances des nœuds hybrides en raison de problèmes d'enregistrement des droits, assurez-vous d'avoir correctement défini votre nom d'utilisateur et votre mot de passe Red Hat sur votre hôte.

```
This system is not registered with an entitlement server
```

Ubuntu

GLIBC introuvable

Si vous utilisez Ubuntu pour votre système d'exploitation et IAM Roles Anywhere pour votre fournisseur d'identifiants avec des nœuds hybrides et que vous constatez qu'un problème avec la GLIBC est introuvable, vous pouvez installer cette dépendance manuellement pour résoudre le problème.

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

Exécutez les commandes suivantes pour installer la dépendance :

```
ldd --version
sudo apt update && apt install libc6
sudo apt install glibc-source
```

Bottlerocket

Si le conteneur d'administration Bottlerocket est activé, vous pouvez y accéder via SSH pour un débogage et un dépannage avancés avec des privilèges élevés. Les sections suivantes contiennent des commandes qui doivent être exécutées dans le contexte de l'hôte Bottlerocket. Une fois que vous êtes sur le conteneur d'administration, vous pouvez exécuter `sheltie` pour obtenir un shell root complet dans l'hôte Bottlerocket.

```
sheltie
```

Vous pouvez également exécuter les commandes des sections suivantes à partir du shell du conteneur d'administration en préfixant chaque commande par`sudo chroot /.bottlerocket/rootfs`.

```
sudo chroot /.bottlerocket/rootfs <command>
```

Utilisation de logdog pour la collecte de logs

Bottlerocket fournit l'`logdog`utilitaire permettant de collecter efficacement les journaux et les informations système à des fins de dépannage.

```
logdog
```

L'`logdog`utilitaire rassemble des journaux provenant de différents emplacements sur un hôte Bottlerocket et les combine dans une archive. Par défaut, l'archive tar est créée à `/var/log/support/bottlerocket-logs.tar.gz` l'adresse et est accessible depuis les conteneurs hôtes à `/.bottlerocket/support/bottlerocket-logs.tar.gz` l'adresse.

Accès aux journaux du système avec journalctl

Vous pouvez vérifier l'état des différents services du système tels que`kubelet`,`containerd`, etc. et consulter leurs journaux à l'aide des commandes suivantes. Le `-f` drapeau suivra les journaux en temps réel.

Pour vérifier l'état du `kubelet` service et récupérer les `kubelet` journaux, vous pouvez exécuter :

```
systemctl status kubelet
journalctl -u kubelet -f
```

Pour vérifier l'état du `containerd` service et récupérer les journaux de l'`containerd`instance orchestrée, vous pouvez exécuter :

```
systemctl status containerd
journalctl -u containerd -f
```

Pour vérifier l'état du `host-containerd` service et récupérer les journaux de l'`containerd`instance hôte, vous pouvez exécuter :

```
systemctl status host-containerd
journalctl -u host-containerd -f
```

Pour récupérer les journaux des conteneurs bootstrap et des conteneurs hôtes, vous pouvez exécuter :

```
journalctl _COMM=host-ctr -f
```

Utiliser le stockage des données d'application pour votre cluster

Vous pouvez utiliser une gamme de services de AWS stockage avec Amazon EKS pour répondre aux besoins de stockage de vos applications. Grâce à un large éventail de pilotes d'interface de stockage de conteneurs (CSI) pris en AWS charge, vous pouvez facilement utiliser Amazon EBS, Amazon S3, Amazon EFS, Amazon FSX et Amazon File Cache pour répondre aux besoins de stockage de vos applications exécutées sur Amazon EKS.

Ce chapitre couvre les options de stockage pour les clusters Amazon EKS.

Rubriques

- Utiliser le stockage en volume Kubernetes avec Amazon EBS
- Utiliser le stockage de système de fichiers élastique avec Amazon EFS
- Utilisez un stockage d'applications performant avec Amazon FSx for Lustre
- Utilisez un stockage d'applications performant avec FSx for NetApp ONTAP
- Utiliser le stockage de données avec Amazon FSx pour OpenZFS
- Minimisez le temps de latence avec Amazon File Cache
- Accédez aux objets Amazon S3 avec le pilote CSI Mountpoint pour Amazon S3
- Activer la fonctionnalité de capture instantanée pour les volumes CSI

Utiliser le stockage en volume Kubernetes avec Amazon EBS

> **Note**
>
> Nouveau : le mode automatique d'Amazon EKS automatise les tâches de routine pour le stockage par blocs. Découvrez comment the section called “Déployez une charge de travail dynamique”.

Le pilote Amazon Elastic Block Store (Amazon EBS) Container Storage Interface (CSI) gère le cycle de vie des volumes Amazon EBS en tant que stockage pour les volumes Kubernetes que vous créez. Le pilote Amazon EBS CSI crée des volumes Amazon EBS pour les types de volumes Kubernetes suivants : volumes éphémères génériques et volumes persistants.

Considérations

- Il n'est pas nécessaire d'installer le contrôleur Amazon EBS CSI sur les clusters en mode automatique EKS.
- Vous ne pouvez pas monter de volumes Amazon EBS sur des Fargate Pods.
- Vous pouvez exécuter le contrôleur Amazon EBS CSI sur des nœuds Fargate, mais le nœud `DaemonSet` Amazon EBS CSI ne peut fonctionner que sur des instances Amazon. EC2
- Les volumes Amazon EBS et le pilote Amazon EBS CSI ne sont pas compatibles avec les nœuds hybrides Amazon EKS.
- Support sera fourni pour la dernière version du module complémentaire et une version précédente. Les bogues ou vulnérabilités découverts dans la dernière version seront rétroportés vers la version précédente dans une nouvelle version mineure.
- Le mode automatique d'EKS nécessite l'utilisation de classes de stockage `ebs.csi.eks.amazonaws.com` en tant que provisionneur. Le pilote Amazon EBS CSI standard (`ebs.csi.aws.com`) gère ses propres volumes séparément. Pour utiliser des volumes existants avec le mode automatique EKS, migrez-les à l'aide de captures d'écran de volume vers une classe de stockage utilisant le provisionneur en mode automatique.

> **Important**
>
> Pour utiliser la fonctionnalité de capture instantanée du pilote Amazon EBS CSI, vous devez d'abord installer le contrôleur de capture instantanée CSI. Pour de plus amples informations, veuillez consulter the section called "Contrôleur d'instantané CSI".

Prérequis

- Un cluster existant. Pour afficher la version requise de la plateforme, exécutez la commande suivante.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- Le pilote EBS CSI nécessite des autorisations AWS IAM.
  - AWS suggère d'utiliser EKS Pod Identities. Pour de plus amples informations, veuillez consulter the section called "Présentation de la configuration des identités du pod EKS".

- Pour plus d'informations sur les rôles IAM pour les comptes de service, consultezthe section called "Fournisseur IAM OIDC".

Étape 1 : Créer un rôle IAM

Le plug-in Amazon EBS CSI nécessite des autorisations IAM pour passer des appels en votre AWS APIs nom. Si vous ne suivez pas ces étapes, la tentative d'installation du module complémentaire et son exécution s'`kubectl describe pvc`afficheront `failed to provision volume with StorageClass` avec un `could not create volume in EC2: UnauthorizedOperation` message d'erreur. Pour plus d'informations, consultez Configuration de l'autorisation du pilote sur GitHub.

> **Note**
>
> Les pods auront accès aux autorisations attribuées au rôle IAM, sauf si vous bloquez l'accès à IMDS. Pour de plus amples informations, veuillez consulter the section called "Bonnes pratiques".

La procédure suivante explique comment créer un rôle IAM et y associer la politique AWS gérée. Pour implémenter cette procédure, vous pouvez utiliser l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

> **Note**
>
> Les étapes spécifiques de cette procédure sont destinées à l'utilisation du pilote en tant que module complémentaire d'Amazon EKS. Différentes étapes sont nécessaires pour utiliser le pilote en tant que module complémentaire autogéré. Pour plus d'informations, voir Configurer les autorisations du conducteur sur GitHub.

eksctl

1. Créez un rôle IAM et associez une politique. AWS gère une politique AWS gérée ou vous pouvez créer votre propre politique personnalisée. Vous pouvez créer un rôle IAM et associer la politique AWS gérée à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster. La commande déploie une AWS CloudFormation pile qui crée un rôle IAM et y attache la politique IAM.

```
eksctl create iamserviceaccount \
        --name ebs-csi-controller-sa \
        --namespace kube-system \
        --cluster my-cluster \
        --role-name AmazonEKS_EBS_CSI_DriverRole \
        --role-only \
        --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
        --approve
```

2. Vous pouvez ignorer cette étape si vous n'utilisez pas de clé KMS personnalisée. Si vous en utilisez un pour le chiffrement de vos volumes Amazon EBS, personnalisez le rôle IAM selon vos besoins. Par exemple, procédez comme suit :

   a. Copiez et collez le code suivant dans un nouveau fichier `kms-key-for-encryption-on-ebs.json`. Remplacez *custom-key-arn* par l'ARN de clé KMS personnalisé.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
```

```
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

b. Créez la politique. Vous pouvez remplacer *KMS_Key_For_Encryption_On_EBS_Policy* par un autre nom. Cependant, si vous le remplacez, assurez-vous de le remplacer également dans les étapes suivantes.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Attachez la politique IAM au rôle à l'aide de la commande suivante. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionnez une entité de confiance), procédez comme suit :
   a. Dans la section Trusted entity type (Type d'entité de confiance), sélectionnez Web identity (Identité web).

b. Pour Fournisseur d'identité, choisissez l'URL du fournisseur OpenID Connect pour votre cluster (comme indiqué sous Présentation dans Amazon EKS).

c. Pour Audience, choisissez `sts.amazonaws.com`.

d. Choisissez Suivant.

5. Sur la page Add permissions (Ajouter des autorisations), procédez comme suit :

   a. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEBSCSIDriverPolicy`.

   b. Cochez la case à gauche du `AmazonEBSCSIDriverPolicy` renvoyé dans la recherche.

   c. Choisissez Suivant.

6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, *`AmazonEKS_EBS_CSI_DriverRole`*.

   b. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

   c. Choisissez Créer un rôle.

7. Une fois le rôle créé, choisissez le rôle dans la console pour l'ouvrir et le modifier.

8. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la politique d'approbation).

9. Trouvez la ligne qui ressemble à la ligne suivante :

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

   Ajoutez une virgule à la fin de la ligne précédente, puis ajoutez la ligne suivante après la ligne précédente. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. *`EXAMPLED539D4633E53DE1B71EXAMPLE`*Remplacez-le par l'ID du fournisseur OIDC de votre cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Sélectionnez Update Trust Policy (Mettre à jour la politique d'approbation) pour terminer.

11. Si vous utilisez une clé KMS personnalisée pour le chiffrement de vos volumes Amazon EBS, personnalisez le rôle IAM en fonction des besoins. Par exemple, procédez comme suit :

    a. Dans le volet de navigation de gauche, choisissez Politiques.

    b. Sur la page Politiques, choisissez Créer une politique.

    c. Sur la page Créer une politique, choisissez l'onglet JSON.

    d. Copiez et collez le code suivant dans l'éditeur, en remplaçant *custom-key-arn* par l'ARN de clé KMS personnalisée.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

    e. Choisissez Suivant : Balises.

    f. Sur la page Ajouter des balises (facultatif), choisissez Suivant : Vérifier.

g. Dans Nom, attribuez un nom unique à votre politique (par exemple *KMS_Key_For_Encryption_On_EBS_Policy*).

h. Choisissez Create Policy (Créer une politique).

i. Dans le volet de navigation de gauche, choisissez Rôles.

j. Choisissez le ***AmazonEKS_EBS_CSI_DriverRole***dans la console pour l'ouvrir en vue de le modifier.

k. Dans la liste déroulante Ajouter des autorisations, choisissez Associer des politiques.

l. Dans la zone Filter policies (Politiques de filtre), saisissez *KMS_Key_For_Encryption_On_EBS_Policy*.

m. Cochez la case à gauche du *KMS_Key_For_Encryption_On_EBS_Policy* renvoyé dans la recherche.

n. Choisissez Attach Policies (Attacher des politiques).

AWS CLI

1. Consultez l'URL du fournisseur OIDC de votre cluster. Remplacez *my-cluster* par le nom de votre cluster. Si la sortie de la commande est `None`, consultez Prérequis.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

L'exemple qui suit illustre un résultat.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Créez le rôle IAM et attachez-y l'action `AssumeRoleWithWebIdentity`.

   a. Copiez le contenu suivant dans un fichier nommé`aws-ebs-csi-driver-trust-policy.json`. Remplacez *111122223333* par votre ID de compte. Remplacez *EXAMPLED539D4633E53DE1B71EXAMPLE* et *region-code* par les valeurs renvoyées par l'étape précédente.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
```

```
          "Principal": {
            "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
          },
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {
            "StringEquals": {
              "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
              "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:ebs-csi-
controller-sa"
            }
          }
        }
      ]
    }
```

b. Créez le rôle. Vous pouvez remplacer *AmazonEKS_EBS_CSI_DriverRole* par un autre nom. Si vous le remplacez, assurez-vous de le remplacer dans les étapes suivantes.

```
aws iam create-role \
      --role-name AmazonEKS_EBS_CSI_DriverRole \
      --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
```

3. Joignez une politique. AWS gère une politique AWS gérée ou vous pouvez créer votre propre politique personnalisée. Attachez la politique AWS gérée au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Si vous utilisez une clé KMS personnalisée pour le chiffrement de vos volumes Amazon EBS, personnalisez le rôle IAM en fonction des besoins. Par exemple, procédez comme suit :

   a. Copiez et collez le code suivant dans un nouveau fichier `kms-key-for-encryption-on-ebs.json`. Remplacez *custom-key-arn* par l'ARN de clé KMS personnalisé.

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
```

```
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["custom-key-arn"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["custom-key-arn"]
        }
      ]
    }
```

b. Créez la politique. Vous pouvez remplacer *KMS_Key_For_Encryption_On_EBS_Policy* par un autre nom. Cependant, si vous le remplacez, assurez-vous de le remplacer également dans les étapes suivantes.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Attachez la politique IAM au rôle à l'aide de la commande suivante. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

Maintenant que vous avez créé le rôle IAM du pilote Amazon EBS CSI, vous pouvez passer à la section suivante. Lorsque vous déployez le module complémentaire avec ce rôle IAM, il crée et est configuré pour utiliser un compte de service nommé`ebs-csi-controller-sa`. Le compte de service est lié à un Kubernetes `clusterrole` auquel les autorisations Kubernetes requises ont été attribuées.

Étape 2 : obtenir le pilote Amazon EBS CSI

Nous vous recommandons d'installer le pilote Amazon EBS CSI via le module complémentaire Amazon EKS afin d'améliorer la sécurité et de réduire la charge de travail. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called "Création d'un module complémentaire". Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

> **Important**
>
> Avant d'ajouter le pilote Amazon EBS en tant que module complémentaire Amazon EKS, vérifiez qu'aucune version autogérée du pilote n'est installée sur votre cluster. Si c'est le cas, voir Désinstallation d'un pilote CSI Amazon EBS autogéré sur GitHub.

Sinon, si vous souhaitez une installation autogérée du pilote Amazon EBS CSI, consultez la section Installation sur. GitHub

Étape 3 : Déployer un exemple d'application

Vous pouvez déployer une variété d'applications types et les modifier selon vos besoins. Pour plus d'informations, consultez les exemples de Kubernetes sur. GitHub

Utiliser le stockage de système de fichiers élastique avec Amazon EFS

Amazon Elastic File System (Amazon EFS) fournit un stockage de fichiers entièrement élastique sans serveur pour vous permettre de partager des données de fichiers sans provisionner ni gérer la capacité et les performances de stockage. Le pilote Amazon EFS Container Storage Interface (CSI) fournit une interface CSI qui permet aux clusters Kubernetes exécutés de gérer le cycle de vie des AWS systèmes de fichiers Amazon EFS. Cette rubrique vous montre comment déployer le pilote Amazon EFS CSI dans votre cluster Amazon EKS.

Considérations

- Le pilote Amazon EFS CSI n'est pas compatible avec les images de conteneur basées sur Windows.
- Vous ne pouvez pas utiliser le provisionnement dynamique pour les volumes persistants avec les nœuds Fargate, mais vous pouvez utiliser le provisionnement statique.
- Le provisionnement dynamique nécessite la version 1.2 ou ultérieure du pilote. Vous pouvez utiliser le provisionnement statique pour les volumes persistants en utilisant la version `1.1` du pilote sur n'importe quelle version de cluster Amazon EKS prise en charge (voir eks/latest/userguide/kubernetes -versions.html [Versions prises en charge par Amazon EKS, type="documentation "]).
- La version 1.3.2 ou ultérieure de ce pilote prend en charge l'architecture Arm64, y compris les instances basées sur Amazon EC2 Graviton.
- La version 1.4.2 ou ultérieure de ce pilote prend en charge l'utilisation de FIPS pour le montage de systèmes de fichiers.
- Prenez note des quotas de ressources pour Amazon EFS. Par exemple, un quota de 1 000 points d'accès peut être créé pour chaque système de fichiers Amazon EFS. Pour plus d'informations, consultez les quotas de ressources Amazon EFS que vous ne pouvez pas modifier.
- À partir de la version 2.0.0, ce pilote est passé de l'utilisation `stunnel` à celle `efs-proxy` des connexions TLS. Lorsqu'il `efs-proxy` est utilisé, il ouvre un nombre de threads égal à un plus le nombre de cœurs pour le nœud sur lequel il s'exécute.
- Le pilote Amazon EFS CSI n'est pas compatible avec les nœuds hybrides Amazon EKS.

Prérequis

- Le pilote Amazon EFS CSI nécessite des autorisations AWS Identity and Access Management (IAM).
  - AWS suggère d'utiliser EKS Pod Identities. Pour de plus amples informations, veuillez consulter the section called “Présentation de la configuration des identités du pod EKS”.
  - Pour plus d'informations sur les rôles IAM pour les comptes de service et sur la configuration d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster, consultez. the section called “Fournisseur IAM OIDC”
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -`

`f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.

- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

> **Note**
>
> Un Pod exécuté sur Fargate monte automatiquement un système de fichiers Amazon EFS, sans nécessiter d'étapes d'installation manuelle du pilote.

Étape 1 : Créer un rôle IAM

Le pilote CSI Amazon EFS nécessite des autorisations IAM pour interagir avec votre système de fichiers. Créez un rôle IAM et associez-y la politique AWS gérée requise. Pour implémenter cette procédure, vous pouvez utiliser l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

> **Note**
>
> Les étapes spécifiques de cette procédure sont destinées à l'utilisation du pilote en tant que module complémentaire d'Amazon EKS. Pour plus de détails sur les installations autogérées, voir Configurer l'autorisation du pilote sur GitHub.

eksctl

Si vous utilisez Pod Identities

Exécutez les commandes suivantes pour créer un rôle IAM et une association avec `eksctl` Pod Identity. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez également le `AmazonEKS_EFS_CSI_DriverRole` remplacer par un autre nom.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create podidentityassociation \
    --service-account-name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --permission-policy-arns arn:aws: iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy
```

Si vous utilisez des rôles IAM pour les comptes de service

Exécutez les commandes suivantes pour créer un rôle IAM avec`eksctl`. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez également le `AmazonEKS_EFS_CSI_DriverRole` remplacer par un autre nom.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

Exécutez la commande suivante pour créer un rôle IAM avec AWS Management Console.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Si vous utilisez EKS Pod Identities :
      i. Dans la section Type d'entité fiable, sélectionnez AWS service.
      ii. Dans le menu déroulant Service ou cas d'utilisation, sélectionnez EKS.
      iii. Dans la section Cas d'utilisation, choisissez EKS - Pod Identity.
      iv. Choisissez Suivant.
   b. Si vous utilisez des rôles IAM pour les comptes de service :
      i. Dans la section Trusted entity type (Type d'entité de confiance), sélectionnez Web identity (Identité web).
      ii. Pour Fournisseur d'identité, choisissez l'URL du fournisseur OpenID Connect pour votre cluster (comme indiqué sous Présentation dans Amazon EKS).
      iii. Pour Audience, choisissez `sts.amazonaws.com`.
      iv. Choisissez Suivant.
5. Sur la page Add permissions (Ajouter des autorisations), procédez comme suit :
   a. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEFSCSIDriverPolicy`.
   b. Cochez la case à gauche du `AmazonEFSCSIDriverPolicy` renvoyé dans la recherche.
   c. Choisissez Suivant.
6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :
   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKS_EFS_CSI_DriverRole`.
   b. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
   c. Choisissez Créer un rôle.
7. Une fois le rôle créé :

a. Si vous utilisez EKS Pod Identities :

   i. Ouvrez la console Amazon EKS.

   ii. Dans le volet de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer l'association EKS Pod Identity.

   iii. Choisissez l'onglet Access.

   iv. Dans Pod Identity associations, choisissez Create.

   v. Choisissez le menu déroulant des rôles IAM et sélectionnez le rôle que vous venez de créer.

   vi. Choisissez le champ et l'entrée de l'espace de noms Kubernetes. `kube-system`

   vii. Choisissez le champ et la saisie du compte de service Kubernetes. `efs-csi-controller-sa`

   viii. Choisissez Créer.

   ix. Pour plus d'informations sur la création d'associations Pod Identity, consultezthe section called "Création d'une association Pod Identity (AWS console)".

b. Si vous utilisez des rôles IAM pour les comptes de service :

   i. Choisissez le rôle pour l'ouvrir en vue de le modifier.

   ii. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la politique d'approbation).

   iii. Trouvez la ligne qui ressemble à la ligne suivante :

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

   Ajoutez la ligne suivante au-dessus de la ligne précédente. Remplacez `<region-code>` par la AWS région dans laquelle se trouve votre cluster. `<EXAMPLED539D4633E53DE1B71EXAMPLE>`Remplacez-le par l'ID du fournisseur OIDC de votre cluster.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

   iv. Modifiez l'opérateur `Condition` de `"StringEquals"` en `"StringLike"`.

   v. Sélectionnez Update Trust Policy (Mettre à jour la politique d'approbation) pour terminer.

AWS CLI

Exécutez les commandes suivantes pour créer un rôle IAM avec la AWS CLI.

Si vous utilisez Pod Identities

1. Créez le rôle IAM qui accorde les `TagSession` actions `AssumeRole` et au `pods.eks.amazonaws.com` service.
   a. Copiez le contenu suivant dans un fichier nommé `aws-efs-csi-driver-trust-policy-pod-identity.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

   b. Créez le rôle. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez également le `AmazonEKS_EFS_CSI_DriverRole` remplacer par un autre nom.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-
identity.json"
```

2. Associez la politique AWS gérée requise au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
```

```
  --role-name $role_name
```

3. Exécutez la commande suivante pour créer l'association Pod Identity. `arn:aws:iam::<111122223333>:role/my-role`Remplacez-le par le rôle créé lors des étapes précédentes.

```
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-aws}iam::<111122223333>:role/my-role --namespace kube-system --service-account efs-csi-controller-sa
```

4. Pour plus d'informations sur la création d'associations Pod Identity, consultezthe section called "Création d'une association Pod Identity (AWS console)".

Si vous utilisez des rôles IAM pour les comptes de service

1. Consultez l'URL du fournisseur OIDC de votre cluster. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez également le `AmazonEKS_EFS_CSI_DriverRole` remplacer par un autre nom.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
 --output text
```

   L'exemple qui suit illustre un résultat.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

   Si la sortie de la commande est `None`, consultez Prérequis.

2. Créez le rôle IAM qui accorde l'action `AssumeRoleWithWebIdentity`.
   a. Copiez le contenu suivant dans un fichier nommé `aws-efs-csi-driver-trust-policy.json`. Remplacez `<111122223333>` par votre ID de compte. Remplacez `<EXAMPLED539D4633E53DE1B71EXAMPLE>` et `<region-code>` par les valeurs renvoyées par l'étape précédente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::<111122223333>:oidc-provider/oidc.eks.<region-
code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
          "oidc.eks.region-code.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. Créez le rôle.

```
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. Associez la politique AWS gérée requise au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

Étape 2 : obtenir le pilote Amazon EFS CSI

Nous vous recommandons d'installer le pilote CSI Amazon EFS en utilisant le module complémentaire Amazon EKS. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called “Création d'un module complémentaire”. Pour plus d'informations sur les modules complémentaires, voir the section called “Modules complémentaires Amazon EKS”. Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

> **Important**
>
> Avant d'ajouter le pilote Amazon EFS en tant que module complémentaire Amazon EKS, vérifiez qu'aucune version autogérée du pilote n'est installée sur votre cluster. Dans ce cas, consultez la section Désinstallation du pilote Amazon EFS CSI sur. GitHub

Sinon, si vous souhaitez une installation autogérée du pilote CSI Amazon EFS, voir Installation sur GitHub.

Étape 3 : Créer un système de fichiers Amazon EFS

Pour créer un système de fichiers Amazon EFS, voir Création d'un système de fichiers Amazon EFS pour Amazon EKS sur GitHub.

Étape 4 : Déployer un exemple d'application

Vous pouvez déployer une variété d'applications types et les modifier selon vos besoins. Pour plus d'informations, consultez la section Exemples sur GitHub.

Utilisez un stockage d'applications performant avec Amazon FSx for Lustre

Le pilote Amazon FSx for Lustre Container Storage Interface (CSI) fournit une interface CSI qui permet aux clusters Amazon EKS de gérer le cycle de vie des systèmes de fichiers Amazon FSx for Lustre. Pour plus d'informations, consultez le guide de l'utilisateur d'Amazon FSx for Lustre.

Pour plus de détails sur la façon de déployer le pilote Amazon FSx for Lustre CSI sur votre cluster Amazon EKS et de vérifier son fonctionnement, consultezthe section called "Déployez le pilote".

Déployez le pilote FSx for Lustre

Cette rubrique explique comment déployer le pilote CSI FSx for Lustre sur votre cluster Amazon EKS et vérifier qu'il fonctionne. Nous vous recommandons d'utiliser la dernière version du pilote. Pour les versions disponibles, consultez la matrice de compatibilité des spécifications CSI sur GitHub.

> **Note**
>
> Le pilote n'est pas pris en charge sur les nœuds hybrides Fargate ou Amazon EKS.

Pour une description détaillée des paramètres disponibles et des exemples complets illustrant les fonctionnalités du pilote, consultez le projet de pilote FSx pour Lustre Container Storage Interface (CSI) sur GitHub.

Prérequis

- Un cluster existant.
- Le module complémentaire Amazon FSx CSI Driver EKS nécessite l'agent EKS Pod Identity pour l'authentification. Sans ce composant, le module complémentaire échouera avec l'erreur`Amazon EKS Pod Identity agent is not installed in the cluster`, empêchant ainsi les opérations de volume. Installez l'agent Pod Identity avant ou après le déploiement du module complémentaire FSx CSI Driver. Pour de plus amples informations, veuillez consulter the section called "Configuration de l'agent".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Version `0.210.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

Étape 1 : Créer un rôle IAM

Le plug-in Amazon FSx CSI nécessite des autorisations IAM pour passer des appels en votre AWS APIs nom.

> **Note**
>
> Les pods auront accès aux autorisations attribuées au rôle IAM, sauf si vous bloquez l'accès à IMDS. Pour de plus amples informations, veuillez consulter the section called "Bonnes pratiques".

La procédure suivante explique comment créer un rôle IAM et y associer la politique AWS gérée.

1. Créez un rôle IAM et associez la politique AWS gérée à l'aide de la commande suivante. `my-cluster`Remplacez-le par le nom du cluster que vous souhaitez utiliser. La commande déploie une AWS CloudFormation pile qui crée un rôle IAM et y attache la politique IAM.

   ```
   eksctl create iamserviceaccount \
       --name fsx-csi-controller-sa \
       --namespace kube-system \
       --cluster my-cluster \
       --role-name AmazonEKS_FSx_CSI_DriverRole \
       --role-only \
       --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
       --approve
   ```

   Vous verrez plusieurs lignes de sortie lors de la création du compte de service. Les dernières lignes de sortie sont similaires à celle de l'exemple suivant.

   ```
   [#]  1 task: {
       2 sequential sub-tasks: {
           create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
           create serviceaccount "kube-system/fsx-csi-controller-sa",
       } }
   [#]  building iamserviceaccount stack "eksctl-my-cluster-addon-iamserviceaccount-
   kube-system-fsx-csi-controller-sa"
   [#]  deploying stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-
   controller-sa"
   [#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-iamserviceaccount-
   kube-system-fsx-csi-controller-sa"
   ```

```
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

Notez le nom de la AWS CloudFormation pile qui a été déployée. Dans l'exemple de sortie ci-dessus, la pile est nommée `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

Maintenant que vous avez créé le rôle IAM du pilote Amazon FSx CSI, vous pouvez passer à la section suivante. Lorsque vous déployez le module complémentaire avec ce rôle IAM, il crée et est configuré pour utiliser un compte de service nommé`fsx-csi-controller-sa`. Le compte de service est lié à un Kubernetes `clusterrole` auquel les autorisations Kubernetes requises ont été attribuées.

Étape 2 : Installation du pilote Amazon FSx CSI

Nous vous recommandons d'installer le pilote Amazon FSx CSI via le module complémentaire Amazon EKS afin d'améliorer la sécurité et de réduire la charge de travail. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called "Création d'un module complémentaire". Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

> **Important**
>
> Les installations préexistantes du pilote Amazon FSx CSI dans le cluster peuvent provoquer l'échec de l'installation des modules complémentaires. Lorsque vous tentez d'installer la version complémentaire Amazon EKS alors qu'un pilote FSx CSI autre qu'EKS existe, l'installation échoue en raison de conflits de ressources. Utilisez le `OVERWRITE` drapeau lors de l'installation pour résoudre ce problème.
>
> ```
> aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>  --resolve-conflicts OVERWRITE
> ```

Sinon, si vous souhaitez une installation autogérée du pilote Amazon FSx CSI, consultez la section Installation sur. GitHub

Étape 3 : déploiement d'une classe de stockage, d'une réclamation de volume persistante et d'un exemple d'application

Cette procédure utilise le GitHub référentiel de pilotes CSI (FSx for Lustre Container Storage Interface) pour utiliser un volume configuré dynamiquement FSx pour Lustre.

1. Notez le groupe de sécurité de votre cluster. Vous pouvez le voir dans la AWS Management Console section Mise en réseau ci-dessous ou en utilisant la commande AWS CLI suivante. `my-cluster`Remplacez-le par le nom du cluster que vous souhaitez utiliser.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. Créez un groupe de sécurité pour votre système de FSx fichiers Amazon conformément aux critères indiqués dans la section Groupes de sécurité Amazon VPC du guide de l'utilisateur Amazon FSx for Lustre. Pour le VPC, sélectionnez le VPC de votre cluster, comme indiqué dans la section Mise en réseau. Pour « les groupes de sécurité associés à vos clients Lustre », utilisez le groupe de sécurité de votre cluster. Vous pouvez juste laisser les règles sortantes pour autoriser Tout le trafic.
3. Téléchargez le manifeste de la classe de stockage à l'aide de la commande suivante.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. Modifiez la section des paramètres du fichier `storageclass.yaml`. Remplacez chaque exemple de valeur par vos propres valeurs.

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **`subnetId`**— L'ID de sous-réseau dans lequel le système de fichiers Amazon FSx for Lustre doit être créé. Amazon FSx for Lustre n'est pas pris en charge dans toutes les zones de disponibilité. Ouvrez la console Amazon FSx for Lustre https://console.aws.amazon.com/fsx/à l'adresse pour vérifier que le sous-réseau que vous souhaitez utiliser se trouve dans une zone de disponibilité prise en charge. Le sous-réseau peut inclure vos nœuds, ou peut être un sous-réseau ou un VPC différent.
  - Vous pouvez vérifier la présence des sous-réseaux de nœuds dans le en AWS Management Console sélectionnant le groupe de nœuds dans la section Calculer.
  - Si le sous-réseau que vous spécifiez n'est pas le même que celui dans lequel vous avez des nœuds, vous VPCs devez être connecté et vous assurer que les ports nécessaires sont ouverts dans vos groupes de sécurité.
- **`securityGroupIds`**— L'ID du groupe de sécurité que vous avez créé pour le système de fichiers.
- **`deploymentType`**(facultatif) — Type de déploiement du système de fichiers. Les valeurs valides sont `SCRATCH_1`, `SCRATCH_2`, `PERSISTENT_1` et `PERSISTENT_2`. Pour plus d'informations sur les types de déploiement, consultez Créer votre système de fichiers Amazon FSx for Lustre.
- autres paramètres (facultatif) — Pour plus d'informations sur les autres paramètres, voir Modifier StorageClass sur GitHub.

5. Créez le manifeste de la classe de stockage.

```
kubectl apply -f storageclass.yaml
```

L'exemple qui suit illustre un résultat.

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Téléchargez le manifeste de revendication de volume persistant.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Facultatif) Modifiez le fichier `claim.yaml`. Remplacez `1200Gi` par l'une des valeurs d'incrément ci-dessous, en fonction de vos besoins de stockage et du `deploymentType` que vous avez sélectionné à l'étape précédente.

```
storage: 1200Gi
```

- SCRATCH_2 et PERSISTENT – 1.2 TiB, 2.4 TiB ou des incréments de 2,4 Tio au-dessus de 2,4 Tio.
- SCRATCH_1 – 1.2 TiB, 2.4 TiB, 3.6 TiB ou des incréments de 3,6 Tio au-dessus de 3,6 Tio.

8. Créez la revendication de volume persistant.

```
kubectl apply -f claim.yaml
```

L'exemple qui suit illustre un résultat.

```
persistentvolumeclaim/fsx-claim created
```

9. Vérifiez que le système de fichiers est approvisionné.

```
kubectl describe pvc
```

L'exemple qui suit illustre un résultat.

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> Le Status peut indiquer Pending pendant 5-10 minutes avant de devenir Bound. Ne passez pas à l'étape suivante tant que Status c'est le casBound. Si le Status affiche Pending pendant plus de 10 minutes, utilisez des messages d'avertissement dans les Events comme référence afin de résoudre tout problème.

10. Déployez un exemple d'application

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-
driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Vérifiez que l'exemple d'application est en cours d'exécution.

```
kubectl get pods
```

L'exemple qui suit illustre un résultat.

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. Vérifiez que le système de fichiers est correctement monté par l'application.

```
kubectl exec -ti fsx-app -- df -h
```

L'exemple qui suit illustre un résultat.

```
Filesystem                   Size  Used Avail Use% Mounted on
overlay                       80G  4.0G   77G   5% /
tmpfs                         64M     0   64M   0% /dev
tmpfs                        3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01      1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1                80G  4.0G   77G   5% /etc/hosts
shm                           64M     0   64M   0% /dev/shm
tmpfs                        6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                        3.8G     0  3.8G   0% /proc/acpi
tmpfs                        3.8G     0  3.8G   0% /sys/firmware
```

13. Vérifiez que les données ont été écrites dans le système de fichiers FSx for Lustre par l'exemple d'application.

```
kubectl exec -it fsx-app -- ls /data
```

L'exemple qui suit illustre un résultat.

```
out.txt
```

Cet exemple de sortie signifie que l'exemple d'application a réussi à écrire le fichier `out.txt` dans le système de fichiers.

> **Note**
>
> Avant de supprimer le cluster, assurez-vous de supprimer le système de fichiers FSx for Lustre. Pour plus d'informations, consultez les ressources de nettoyage dans le guide de l'utilisateur de FSx for Lustre.

Optimisation des performances FSx pour Lustre

Lorsque vous utilisez FSx for Lustre avec Amazon EKS, vous pouvez optimiser les performances en appliquant les réglages Lustre lors de l'initialisation du nœud. L'approche recommandée consiste à utiliser les données utilisateur du modèle de lancement pour garantir une configuration cohérente sur tous les nœuds.

Ces réglages incluent :

- Optimisations du réseau et du RPC
- Gestion du module Lustre
- Réglages LRU (Lock Resource Unit)
- Paramètres de contrôle du cache client
- Contrôles RPC pour OST et MDC

Pour obtenir des instructions détaillées sur la mise en œuvre de ces réglages de performance :

- Pour optimiser les performances des nœuds standard (non EFA), consultez the section called "Optimize (non-EFA)" un script complet qui peut être ajouté aux données utilisateur de votre modèle de lancement.
- Pour optimiser les performances des nœuds compatibles EFA, consultez. the section called "Optimiser (EFA)"

Optimisez les performances FSx d'Amazon for Lustre sur les nœuds (EFA)

Cette rubrique explique comment configurer le réglage d'Elastic Fabric Adapter (EFA) avec Amazon EKS et FSx Amazon for Lustre.

> **Note**
>
> - Pour plus d'informations sur la création et le déploiement du pilote CSI FSx for Lustre, consultezthe section called "Déployez le pilote".
> - Pour optimiser les nœuds standard sans EFA, voirthe section called "Optimize (non-EFA)".

Étape 1. Création d'un cluster EKS

Créez un cluster à l'aide du fichier de configuration fourni :

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

Exemple `efa-cluster.yaml` :

```
#efa-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: csi-fsx
  region: us-east-1
  version: "1.30"

iam:
  withOIDC: true

availabilityZones: ["us-east-1a", "us-east-1d"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
```

```
    maxSize: 1
    availabilityZones: ["us-east-1b"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print $1}')"
        efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//g')
        min_efa_version="2.12.1"

        if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
            sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
            tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
            echo "Installing EFA driver"
            sudo apt-get update && apt-get upgrade -y
            sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms
            sudo ./efa_installer.sh -y
            modinfo efa
        else
            echo "Using EFA driver version $efa_version"
        fi

        echo "Installing Lustre client"
        sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-
key.gpg > /dev/null
        sudo echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/
sources.list.d/fsxlustreclientrepo.list
        sudo apt update | tail
        sudo apt install -y lustre-client-modules-$(uname -r) amazon-ec2-utils | tail
        modinfo lustre

        echo "Loading Lustre/EFA modules..."
        sudo /sbin/modprobe lnet
        sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
        sudo /sbin/modprobe ksocklnd
        sudo lnetctl lnet configure
```

```
        echo "Configuring TCP interface..."
        sudo lnetctl net del --net tcp 2> /dev/null
        sudo lnetctl net add --net tcp --if $eth_intf

        # For P5 instance type which supports 32 network cards,
        # by default add 8 EFA interfaces selecting every 4th device (1 per PCI bus)
        echo "Configuring EFA interface(s)..."
        instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
        num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
        echo "Found $num_efa_devices available EFA device(s)"

        if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
 "p5e.48xlarge" ]]; then
           for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
               sudo lnetctl net add --net efa --if $intf --peer-credits 32
          done
        else
        # Other instances: Configure 2 EFA interfaces by default if the instance
 supports multiple network cards,
        # or 1 interface for single network card instances
        # Can be modified to add more interfaces if instance type supports it
            sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | head -
n1) --peer-credits 32
            if [[ $num_efa_devices -gt 1 ]]; then
               sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | tail
 -n1) --peer-credits 32
            fi
        fi

        echo "Setting discovery and UDSP rule"
        sudo lnetctl set discovery 1
        sudo lnetctl udsp add --src efa --priority 0
        sudo /sbin/modprobe lustre

        sudo lnetctl net show
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
```

Étape 2. Créer un groupe de nœuds

Créez un groupe de nœuds compatible EFA :

```
# Create node group using efa-ng.yaml
```

```
eksctl create nodegroup -f efa-ng.yaml
```

> **Important**
>
> === Ajustez ces valeurs pour votre environnement dans la section`# 5. Mount FSx filesystem`.

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

===

Exemple `efa-ng.yaml` :

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: final-efa
  region: us-east-1

managedNodeGroups:
  - name: ng-1
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1a"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        exec 1> >(logger -s -t $(basename $0)) 2>&1


 ##############################################################################
```

```
        #                                   Configuration Section
         #

 ##########################################################################################

        # File System Configuration
        FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
        MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
        MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.

        # Lustre Tuning Parameters
        LUSTRE_LRU_MAX_AGE=600000
        LUSTRE_MAX_CACHED_MB=64
        LUSTRE_OST_MAX_RPC=32
        LUSTRE_MDC_MAX_RPC=64
        LUSTRE_MDC_MOD_RPC=50

        # File paths
        FUNCTIONS_SCRIPT="/usr/local/bin/lustre_functions.sh"
        TUNINGS_SCRIPT="/usr/local/bin/apply_lustre_tunings.sh"
        SERVICE_FILE="/etc/systemd/system/lustre-tunings.service"

        #EFA
        MIN_EFA_VERSION="2.12.1"

        # Function to check if a command was successful
        check_success() {
            if [ $? -eq 0 ]; then
                echo "SUCCESS: $1"
            else
                echo "FAILED: $1"
                return 1
            fi
        }

        echo "********Starting FSx for Lustre configuration********"

        # 1. Install Lustre client
        if grep -q '^ID=ubuntu' /etc/os-release; then
            echo "Detected Ubuntu, proceeding with Lustre setup..."
            # Add Lustre repository
            sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/
fsx-ubuntu-public-key.asc | sudo gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg > /dev/null
```

```
            echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" | sudo tee /etc/
apt/sources.list.d/fsxlustreclientrepo.list

            sudo apt-get update
            sudo apt-get install -y lustre-client-modules-$(uname -r)
            sudo apt-get install -y lustre-client
        else
            echo "Not Ubuntu, exiting"
            exit 1
        fi

        check_success "Install Lustre client"

        # Ensure Lustre tools are in the PATH
        export PATH=$PATH:/usr/sbin

        # 2. Apply network and RPC tunings
        echo "********Applying network and RPC tunings********"
        if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/
modprobe.conf; then
            echo "options ptlrpc ptlrpcd_per_cpt_max=64" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ptlrpcd_per_cpt_max"
        else
            echo "ptlrpcd_per_cpt_max already set in modprobe.conf"
        fi

        if ! grep -q "options ksocklnd credits" /etc/modprobe.d/modprobe.conf; then
            echo "options ksocklnd credits=2560" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ksocklnd credits"
        else
            echo "ksocklnd credits already set in modprobe.conf"
        fi

        # 3. Load Lustre modules
        manage_lustre_modules() {
            echo "Checking for existing Lustre modules..."
            if lsmod | grep -q lustre; then
                echo "Existing Lustre modules found."

                # Check for mounted Lustre filesystems
```

```
                echo "Checking for mounted Lustre filesystems..."
                if mount | grep -q "type lustre"; then
                    echo "Found mounted Lustre filesystems. Attempting to unmount..."
                    mounted_fs=$(mount | grep "type lustre" | awk '{print $3}')
                    for fs in $mounted_fs; do
                        echo "Unmounting $fs"
                        sudo umount $fs
                        check_success "Unmount filesystem $fs"
                    done
                else
                    echo "No Lustre filesystems mounted."
                fi

                # After unmounting, try to remove modules
                echo "Attempting to remove Lustre modules..."
                sudo lustre_rmmod
                if [ $? -eq 0 ]; then
                    echo "SUCCESS: Removed existing Lustre modules"
                else
                    echo "WARNING: Could not remove Lustre modules. They may still be
 in use."
                    echo "Please check for any remaining Lustre processes or mounts."
                    return 1
                fi
            else
                echo "No existing Lustre modules found."
            fi

            echo "Loading Lustre modules..."
            sudo modprobe lustre
            check_success "Load Lustre modules" || exit 1

            echo "Checking loaded Lustre modules:"
            lsmod | grep lustre
        }

        # Managing Lustre kernel modules
        echo "********Managing Lustre kernel modules********"
        manage_lustre_modules

        # 4. Initializing Lustre networking
        echo "********Initializing Lustre networking********"
        sudo lctl network up
        check_success "Initialize Lustre networking" || exit 1
```

```
        # 4.5 EFA Setup and Configuration
        setup_efa() {
            echo "********Starting EFA Setup********"

            # Get interface and version information
            eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print
 $1}')"
            efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//
g')
            min_efa_version=$MIN_EFA_VERSION

            # Install or verify EFA driver
            if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
                echo "Installing EFA driver..."
                sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
                tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer

                # Install dependencies
                sudo apt-get update && apt-get upgrade -y
                sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms

                # Install EFA
                sudo ./efa_installer.sh -y
                modinfo efa
            else
                echo "Using existing EFA driver version $efa_version"
            fi
        }

        configure_efa_network() {
            echo "********Configuring EFA Network********"

            # Load required modules
            echo "Loading network modules..."
            sudo /sbin/modprobe lnet
            sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
            sudo /sbin/modprobe ksocklnd

            # Initialize LNet
            echo "Initializing LNet..."
```

```
            sudo lnetctl lnet configure

            # Configure TCP interface
            echo "Configuring TCP interface..."
            sudo lnetctl net del --net tcp 2> /dev/null
            sudo lnetctl net add --net tcp --if $eth_intf

            # For P5 instance type which supports 32 network cards,
            # by default add 8 EFA interfaces selecting every 4th device (1 per PCI
bus)
            echo "Configuring EFA interface(s)..."
            instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
            num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
            echo "Found $num_efa_devices available EFA device(s)"

            if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
"p5e.48xlarge" ]]; then
                # P5 instance configuration
                for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
                    sudo lnetctl net add --net efa --if $intf --peer-credits 32
                done
            else
                # Standard configuration
                # Other instances: Configure 2 EFA interfaces by default if the
instance supports multiple network cards,
                # or 1 interface for single network card instances
                # Can be modified to add more interfaces if instance type supports it
                sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
head -n1) --peer-credits 32
                if [[ $num_efa_devices -gt 1 ]]; then
                    sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
tail -n1) --peer-credits 32
                fi
            fi

            # Configure discovery and UDSP
            echo "Setting up discovery and UDSP rules..."
            sudo lnetctl set discovery 1
            sudo lnetctl udsp add --src efa --priority 0
            sudo /sbin/modprobe lustre

            # Verify configuration
            echo "Verifying EFA network configuration..."
            sudo lnetctl net show
```

```
            echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
        }

        # Main execution
        setup_efa
        configure_efa_network

        # 5. Mount FSx filesystem
        if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
            echo "********Creating mount point********"
            sudo mkdir -p $MOUNT_POINT
            check_success "Create mount point"

            echo "Mounting FSx filesystem..."
            sudo mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
            check_success "Mount FSx filesystem"
        else
            echo "Skipping FSx mount as DNS or mount name is not provided"
        fi

        # 6. Applying Lustre performance tunings
        echo "********Applying Lustre performance tunings********"

        # Get number of CPUs
        NUM_CPUS=$(nproc)

        # Calculate LRU size (100 * number of CPUs)
        LRU_SIZE=$((100 * NUM_CPUS))

        #Apply LRU tunings
        echo "Apply LRU tunings"
        sudo lctl set_param ldlm.namespaces.*.lru_max_age=${LUSTRE_LRU_MAX_AGE}
        check_success "Set lru_max_age"
        sudo lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
        check_success "Set lru_size"

        # Client Cache Control
        sudo lctl set_param llite.*.max_cached_mb=${LUSTRE_MAX_CACHED_MB}
        check_success "Set max_cached_mb"

        # RPC Controls
        sudo lctl set_param osc.*OST*.max_rpcs_in_flight=${LUSTRE_OST_MAX_RPC}
        check_success "Set OST max_rpcs_in_flight"
```

```
        sudo lctl set_param mdc.*.max_rpcs_in_flight=${LUSTRE_MDC_MAX_RPC}
        check_success "Set MDC max_rpcs_in_flight"

        sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=${LUSTRE_MDC_MOD_RPC}
        check_success "Set MDC max_mod_rpcs_in_flight"

        # 7. Verify all tunings
        echo "********Verifying all tunings********"

        # Function to verify parameter value
        verify_param() {
            local param=$1
            local expected=$2
            local actual=$3

            if [ "$actual" == "$expected" ]; then
                echo "SUCCESS: $param is correctly set to $expected"
            else
                echo "WARNING: $param is set to $actual (expected $expected)"
            fi
        }

        echo "Verifying all parameters:"

        # LRU tunings
        actual_lru_max_age=$(lctl get_param -n ldlm.namespaces.*.lru_max_age | head -1)
        verify_param "lru_max_age" "600000" "$actual_lru_max_age"

        actual_lru_size=$(lctl get_param -n ldlm.namespaces.*.lru_size | head -1)
        verify_param "lru_size" "$LRU_SIZE" "$actual_lru_size"

        # Client Cache
        actual_max_cached_mb=$(lctl get_param -n llite.*.max_cached_mb | grep
 "max_cached_mb:" | awk '{print $2}')
        verify_param "max_cached_mb" "64" "$actual_max_cached_mb"

        # RPC Controls
        actual_ost_rpcs=$(lctl get_param -n osc.*OST*.max_rpcs_in_flight | head -1)
        verify_param "OST max_rpcs_in_flight" "32" "$actual_ost_rpcs"

        actual_mdc_rpcs=$(lctl get_param -n mdc.*.max_rpcs_in_flight | head -1)
        verify_param "MDC max_rpcs_in_flight" "64" "$actual_mdc_rpcs"

        actual_mdc_mod_rpcs=$(lctl get_param -n mdc.*.max_mod_rpcs_in_flight | head -1)
```

```
        verify_param "MDC max_mod_rpcs_in_flight" "50" "$actual_mdc_mod_rpcs"

        # Network and RPC configurations from modprobe.conf
        actual_ptlrpc=$(grep "ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf
| awk '{print $3}')
        verify_param "ptlrpcd_per_cpt_max" "ptlrpcd_per_cpt_max=64" "$actual_ptlrpc"

        actual_ksocklnd=$(grep "ksocklnd credits" /etc/modprobe.d/modprobe.conf | awk
'{print $3}')
        verify_param "ksocklnd credits" "credits=2560" "$actual_ksocklnd"

        # 8. Setup persistence
        setup_persistence() {
            # Create functions file
            cat << EOF > $FUNCTIONS_SCRIPT
        #!/bin/bash

        apply_lustre_tunings() {
            local NUM_CPUS=\$(nproc)
            local LRU_SIZE=\$((100 * NUM_CPUS))

            echo "Applying Lustre performance tunings..."
            lctl set_param ldlm.namespaces.*.lru_max_age=$LUSTRE_LRU_MAX_AGE
            lctl set_param ldlm.namespaces.*.lru_size=\$LRU_SIZE
            lctl set_param llite.*.max_cached_mb=$LUSTRE_MAX_CACHED_MB
            lctl set_param osc.*OST*.max_rpcs_in_flight=$LUSTRE_OST_MAX_RPC
            lctl set_param mdc.*.max_rpcs_in_flight=$LUSTRE_MDC_MAX_RPC
            lctl set_param mdc.*.max_mod_rpcs_in_flight=$LUSTRE_MDC_MOD_RPC
        }
        EOF

            # Create tuning script
            cat << EOF > $TUNINGS_SCRIPT
        #!/bin/bash
        exec 1> >(logger -s -t \$(basename \$0)) 2>&1

        source $FUNCTIONS_SCRIPT

        # Function to check if Lustre is mounted
        is_lustre_mounted() {
            mount | grep -q "type lustre"
        }

        # Function to mount Lustre
```

```
        mount_lustre() {
            echo "Mounting Lustre filesystem..."
            mkdir -p $MOUNT_POINT
            mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} $MOUNT_POINT
            return \$?
        }

        # Main execution
        # Try to mount if not already mounted
        if ! is_lustre_mounted; then
            echo "Lustre filesystem not mounted, attempting to mount..."
            mount_lustre
        fi

        # Wait for successful mount (up to 5 minutes)
        for i in {1..30}; do
            if is_lustre_mounted; then
                echo "Lustre filesystem mounted, applying tunings..."
                apply_lustre_tunings
                exit 0
            fi
            echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
            sleep 10
        done

        echo "Timeout waiting for Lustre filesystem mount"
        exit 1
        EOF

        # Create systemd service

        # Create systemd directory if it doesn't exist
        sudo mkdir -p /etc/systemd/system/

            # Create service file directly for Ubuntu
            cat << EOF > $SERVICE_FILE
        [Unit]
        Description=Apply Lustre Performance Tunings
        After=network.target remote-fs.target

        [Service]
        Type=oneshot
        ExecStart=/bin/bash -c 'source $FUNCTIONS_SCRIPT && $TUNINGS_SCRIPT'
        RemainAfterExit=yes
```

```
        [Install]
        WantedBy=multi-user.target
        EOF


            # Make scripts executable and enable service
            sudo chmod +x $FUNCTIONS_SCRIPT
            sudo chmod +x $TUNINGS_SCRIPT
            systemctl enable lustre-tunings.service
            systemctl start lustre-tunings.service
        }

        echo "********Setting up persistent tuning********"
        setup_persistence

        echo "FSx for Lustre configuration completed."
```

(Facultatif) Étape 3. Vérifiez la configuration de l'EFA

SSH dans le nœud :

```
# Get instance ID from EKS console or AWS CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

Vérifiez la configuration EFA :

```
sudo lnetctl net show
```

Consultez les journaux de configuration :

```
sudo cat /var/log/cloud-init-output.log
```

Voici un exemple de sortie attendue pour `lnetctl net show` :

```
net:
    - net type: tcp
      ...
    - net type: efa
      local NI(s):
        - nid: xxx.xxx.xxx.xxx@efa
```

```
          status: up
```

Exemples de déploiements

a. Créez claim.yaml

```
#claim.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fsx-claim-efa
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 4800Gi
  volumeName: fsx-pv
```

Appliquez la réclamation :

```
kubectl apply -f claim.yaml
```

b. Créez pv.yaml

Mettez à jour le `<replaceable-placeholders>` :

```
#pv.yaml

apiVersion: v1
kind: PersistentVolume
metadata:
  name: fsx-pv
spec:
  capacity:
    storage: 4800Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  mountOptions:
```

```
    - flock
  persistentVolumeReclaimPolicy: Recycle
  csi:
    driver: fsx.csi.aws.com
    volumeHandle: fs-<1234567890abcdef0>
    volumeAttributes:
      dnsname: fs-<1234567890abcdef0>.fsx.us-east-1.amazonaws.com
      mountname: <abcdef01>
```

Appliquez le volume persistant :

```
kubectl apply -f pv.yaml
```

c. Créez pod.yaml

```
#pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: fsx-efa-app
spec:
  containers:
  - name: app
    image: amazonlinux:2
    command: ["/bin/sh"]
    args: ["-c", "while true; do dd if=/dev/urandom bs=100M count=20 > data/test_file;
 sleep 10; done"]
    resources:
      requests:
        vpc.amazonaws.com/efa: 1
      limits:
        vpc.amazonaws.com/efa: 1
    volumeMounts:
    - name: persistent-storage
      mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: fsx-claim-efa
```

Appliquez le pod :

```
kubectl apply -f pod.yaml
```

Commandes de vérification supplémentaires

Vérifiez que Pod monte et écrit sur le système de fichiers :

```
kubectl exec -ti fsx-efa-app -- df -h | grep data
# Expected output:
# <192.0.2.0>@tcp:/<abcdef01>  4.5T  1.2G  4.5T   1% /data

kubectl exec -ti fsx-efa-app -- ls /data
# Expected output:
# test_file
```

Connectez-vous au nœud en SSH pour vérifier que le trafic passe par EFA :

```
sudo lnetctl net show -v
```

Le résultat attendu affichera les interfaces EFA avec les statistiques de trafic.

Informations connexes

- the section called "Déployez le pilote"
- the section called "Optimize (non-EFA)"
- Amazon FSx pour Lustre Performance
- Elastic Fabric Adapter (EFA)

Optimisez les performances FSx d'Amazon for Lustre sur les nœuds (non EFA)

Vous pouvez optimiser les performances d'Amazon FSx for Lustre en appliquant des paramètres de réglage lors de l'initialisation du nœud à l'aide des données utilisateur du modèle de lancement.

> **Note**
>
> - Pour plus d'informations sur la création et le déploiement du pilote CSI FSx for Lustre, consultezthe section called "Déployez le pilote". Pour optimiser les performances avec les nœuds compatibles EFA, consultez. the section called "Optimiser (EFA)"

Pourquoi utiliser les données utilisateur du modèle de lancement ?

- Applique les réglages automatiquement lors de l'initialisation du nœud.
- Garantit une configuration cohérente sur tous les nœuds.
- Élimine le besoin de configuration manuelle des nœuds.

Présentation d'un exemple de script

L'exemple de script défini dans cette rubrique effectue les opérations suivantes :

`# 1. Install Lustre client`

- Détecte automatiquement la version de votre système d'exploitation : Amazon Linux 2 (AL2) ou Amazon Linux 2023 (AL2023).
- Installe le package client Lustre approprié.

`# 2. Apply network and RPC tunings`

- Ensembles `ptlrpcd_per_cpt_max=64` pour le traitement RPC parallèle.
- Configure `ksocklnd credits=2560` pour optimiser les tampons réseau.

`# 3. Load Lustre modules`

- Supprime en toute sécurité les modules Lustre existants s'ils sont présents.
- Gère le démontage des systèmes de fichiers existants.
- Charge de nouveaux modules Lustre.

4. Lustre Network Initialization

- Initialise la configuration réseau Lustre.
- Configure les paramètres réseau requis.

5. Mount FSx filesystem

- Vous devez ajuster les valeurs de votre environnement dans cette section.

6. Apply tunings

- Réglages LRU (Lock Resource Unit) :
  - `lru_max_age=600000`
  - `lru_size`calculé en fonction du nombre de processeurs
- Contrôle du cache client : `max_cached_mb=64`
- Contrôles RPC :
  - OST `max_rpcs_in_flight=32`
  - MDC `max_rpcs_in_flight=64`
  - MDC `max_mod_rpcs_in_flight=50`

7. Verify tunings

- Vérifie tous les réglages appliqués.
- Signale le succès ou un avertissement pour chaque paramètre.

8. Setup persistence

- Vous devez également ajuster les valeurs de votre environnement dans cette section.
- Détecte automatiquement la version de votre système d'exploitation (AL2 ou AL2 023) pour déterminer le `Systemd` service à appliquer.
- Le système démarre.
- `Systemd`démarre `lustre-tunings` le service (en raison de`WantedBy=multi-user.target`).
- Un service fonctionne `apply_lustre_tunings.sh` qui :
  - Vérifie si le système de fichiers est monté.

  - Monte le système de fichiers s'il n'est pas monté.
  - Attend un montage réussi (jusqu'à cinq minutes).
  - Applique les paramètres de réglage après un montage réussi.
- Les paramètres restent actifs jusqu'au redémarrage.
- Le service se ferme une fois le script terminé.
  - Systemd marque le service comme « actif (terminé) ».
- Le processus se répète au prochain redémarrage.

Création d'un modèle de lancement

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Launch Templates.
3. Choisissez Créer un modèle de lancement.
4. Dans Détails avancés, recherchez la section Données utilisateur.
5. Collez le script ci-dessous, en mettant à jour tout ce qui est nécessaire.

> **Important**
>
> Ajustez ces valeurs en fonction de votre environnement dans la section `# 5. Mount FSx filesystem` et de la `setup_persistence()` fonction `apply_lustre_tunings.sh` de la section `# 8. Setup persistence` :
>
> ```
> FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
> MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
> MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
> ```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Function definitions
check_success() {
```

```
    if [ $? -eq 0 ]; then
        echo "SUCCESS: $1"
    else
        echo "FAILED: $1"
        return 1
    fi
}
apply_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age=600000"
        "ldlm.namespaces.*.lru_size=$LRU_SIZE"
        "llite.*.max_cached_mb=64"
        "osc.*OST*.max_rpcs_in_flight=32"
        "mdc.*.max_rpcs_in_flight=64"
        "mdc.*.max_mod_rpcs_in_flight=50"
    )
    for param in "${params[@]}"; do
        lctl set_param $param
        check_success "Set ${param%%=*}"
    done
}
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}
verify_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age:600000"
        "ldlm.namespaces.*.lru_size:$LRU_SIZE"
        "llite.*.max_cached_mb:64"
        "osc.*OST*.max_rpcs_in_flight:32"
        "mdc.*.max_rpcs_in_flight:64"
        "mdc.*.max_mod_rpcs_in_flight:50"
```

```
    )
    echo "Verifying all parameters:"
    for param in "${params[@]}"; do
        name="${param%%:*}"
        expected="${param#*:}"
        actual=$(lctl get_param -n $name | head -1)
        verify_param "${name##*.}" "$expected" "$actual"
    done
}
setup_persistence() {
    # Create functions file
    cat << 'EOF' > /usr/local/bin/lustre_functions.sh
#!/bin/bash
apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldlm.namespaces.*.lru_max_age=600000
    lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=64
    lctl set_param osc.*OST*.max_rpcs_in_flight=32
    lctl set_param mdc.*.max_rpcs_in_flight=64
    lctl set_param mdc.*.max_mod_rpcs_in_flight=50
}
EOF
    # Create tuning script
    cat << 'EOF' > /usr/local/bin/apply_lustre_tunings.sh
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Source the functions
source /usr/local/bin/lustre_functions.sh
# FSx details
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"
}
# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p ${MOUNT_POINT}
```

```
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    return $?
}
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi
# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done
echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF
    # Create systemd service
    cat << 'EOF' > /etc/systemd/system/lustre-tunings.service
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target
StartLimitIntervalSec=0
[Service]
Type=oneshot
ExecStart=/usr/local/bin/apply_lustre_tunings.sh
RemainAfterExit=yes
Restart=on-failure
RestartSec=30
[Install]
WantedBy=multi-user.target
EOF
    chmod +x /usr/local/bin/lustre_functions.sh
    chmod +x /usr/local/bin/apply_lustre_tunings.sh
    systemctl enable lustre-tunings.service
    systemctl start lustre-tunings.service
}
echo "Starting FSx for Lustre configuration..."
# 1. Install Lustre client
```

```
if grep -q 'VERSION="2"' /etc/os-release; then
    amazon-linux-extras install -y lustre
elif grep -q 'VERSION="2023"' /etc/os-release; then
    dnf install -y lustre-client
fi
check_success "Install Lustre client"
# 2. Apply network and RPC tunings
export PATH=$PATH:/usr/sbin
echo "Applying network and RPC tunings..."
if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf; then
    echo "options ptlrpc ptlrpcd_per_cpt_max=64" | tee -a /etc/modprobe.d/
modprobe.conf
    echo "options ksocklnd credits=2560" | tee -a /etc/modprobe.d/modprobe.conf
fi
# 3. Load Lustre modules
modprobe lustre
check_success "Load Lustre modules" || exit 1
# 4. Lustre Network Initialization
lctl network up
check_success "Initialize Lustre networking" || exit 1
# 5. Mount FSx filesystem
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
fi
# 6. Apply tunings
apply_tunings
# 7. Verify tunings
verify_tunings
# 8. Setup persistence
setup_persistence
echo "FSx for Lustre configuration completed."
--==MYBOUNDARY==--
```

6. Lorsque vous créez des groupes de nœuds Amazon EKS, sélectionnez ce modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Création".

Informations connexes

- the section called "Déployez le pilote"
- the section called "Optimiser (EFA)"
- Amazon FSx pour Lustre Performance

Utilisez un stockage d'applications performant avec FSx for NetApp ONTAP

Le NetApp Trident assure une orchestration dynamique du stockage à l'aide d'un pilote compatible CSI (Container Storage Interface). Cela permet aux clusters Amazon EKS de gérer le cycle de vie des volumes persistants (PVs) soutenus par Amazon FSx pour les systèmes de fichiers NetApp ONTAP. Notez que le pilote Amazon FSx for NetApp ONTAP CSI n'est pas compatible avec les nœuds hybrides Amazon EKS. Pour commencer, consultez la section Utiliser Trident avec Amazon FSx pour NetApp ONTAP dans la documentation de NetApp Trident.

Amazon FSx for NetApp ONTAP est un service de stockage qui vous permet de lancer et d'exécuter des systèmes de fichiers ONTAP entièrement gérés dans le cloud. ONTAP NetApp est une technologie de système de fichiers qui fournit un ensemble largement adopté de fonctionnalités d'accès et de gestion des données. FSx for ONTAP fournit les fonctionnalités, les performances et les systèmes APIs de NetApp fichiers sur site avec l'agilité, l'évolutivité et la simplicité d'un service entièrement géré. AWS Pour plus d'informations, consultez le guide de FSx l'utilisateur d'ONTAP.

> **Important**
>
> Si vous utilisez Amazon FSx for NetApp ONTAP avec le pilote Amazon EBS CSI pour approvisionner des volumes EBS, vous devez spécifier de ne pas utiliser de périphériques EBS dans le fichier. `multipath.conf` Pour connaître les méthodes prises en charge, consultez la section Liste noire des fichiers de configuration. Voici un exemple.

```
defaults {
       user_friendly_names yes
       find_multipaths no
     }
     blacklist {
       device {
         vendor "NVME"
```

```
          product "Amazon Elastic Block Store"
        }
      }
```

Utiliser le stockage de données avec Amazon FSx pour OpenZFS

Amazon FSx pour OpenZFS est un service de stockage de fichiers entièrement géré qui facilite le transfert de données depuis des serveurs de fichiers ZFS sur site ou AWS d'autres serveurs de fichiers basés sur Linux. Vous pouvez le faire sans modifier le code de votre application ni la façon dont vous gérez les données. Il offre un stockage de fichiers hautement fiable, évolutif, efficace et riche en fonctionnalités basé sur le système de fichiers open source OpenZFS. Il associe ces fonctionnalités à l'agilité, à l'évolutivité et à la simplicité d'un AWS service entièrement géré. Pour plus d'informations, consultez le guide de l'utilisateur FSx d'Amazon pour OpenZFS.

Le pilote CSI (Container Storage Interface) FSx pour OpenZFS fournit une interface CSI qui permet aux clusters Amazon EKS de gérer le cycle de vie des FSx volumes OpenZFS. Notez que le pilote Amazon FSx pour OpenZFS CSI n'est pas compatible avec les nœuds hybrides Amazon EKS. Pour déployer le pilote CSI FSx pour OpenZFS sur votre cluster Amazon EKS, consultez aws-fsx-openzfs-csi -driver on. GitHub

Minimisez le temps de latence avec Amazon File Cache

Amazon File Cache est un cache haut débit entièrement géré AWS qui est utilisé pour traiter les données des fichiers, quel que soit leur emplacement de stockage. Amazon File Cache charge automatiquement les données dans le cache lors du premier accès et publie les données lorsqu'elles ne sont pas utilisées. Pour de plus amples informations, veuillez consulter le Guide de l'utilisateur Amazon File Cache.

Le pilote Amazon File Cache Container Storage Interface (CSI) fournit une interface CSI qui permet aux clusters Amazon EKS de gérer le cycle de vie des systèmes de fichiers Amazon. Notez que le pilote Amazon File Cache CSI n'est pas compatible avec les nœuds hybrides Amazon EKS. Pour déployer le pilote Amazon File Cache CSI sur votre cluster Amazon EKS, consultez aws-file-cache-csi-driver on. GitHub

Accédez aux objets Amazon S3 avec le pilote CSI Mountpoint pour Amazon S3

Avec le pilote Mountpoint for Amazon S3 Container Storage Interface (CSI), vos applications Kubernetes peuvent accéder aux objets Amazon S3 via une interface de système de fichiers, ce qui permet d'obtenir un débit agrégé élevé sans modifier le code de l'application. Construit sur Mountpoint pour Amazon S3, le pilote CSI présente un compartiment Amazon S3 sous la forme d'un volume accessible par des conteneurs dans Amazon EKS et des clusters Kubernetes autogérés.

Considérations

- Le pilote CSI Mountpoint pour Amazon S3 n'est actuellement pas compatible avec les images de conteneur basées sur Windows.
- Le pilote CSI Mountpoint pour Amazon S3 n'est actuellement pas compatible avec les nœuds hybrides Amazon EKS.
- Le pilote CSI Mountpoint pour Amazon S3 ne prend pas en charge Fargate. AWS Toutefois, les conteneurs qui s'exécutent dans Amazon EC2 (soit avec Amazon EKS, soit avec une installation Kubernetes personnalisée) sont pris en charge.
- Le pilote CSI Mountpoint pour Amazon S3 ne prend en charge que le provisionnement statique. Le provisionnement dynamique ou la création de nouveaux compartiments ne sont pas pris en charge.

> **Note**
>
> Le provisionnement statique fait référence à l'utilisation d'un compartiment Amazon S3 existant spécifié comme étant `bucketName` `volumeAttributes` dans l'`PersistentVolume`objet. Pour plus d'informations, consultez Static Provisioning on GitHub.

- Les volumes montés avec le pilote CSI Mountpoint pour Amazon S3 ne prennent pas en charge toutes les fonctionnalités du système de fichiers POSIX. Pour plus d'informations sur le comportement du système de fichiers, consultez Mountpoint pour le comportement du système de fichiers Amazon S3 sur. GitHub

Pour plus de détails sur le déploiement du pilote, consultezthe section called “Déployez le pilote”. Pour plus de détails sur le retrait du pilote, consultezthe section called “Supprimer le pilote”.

Déployez le pilote Mountpoint pour Amazon S3

Avec le pilote Mountpoint for Amazon S3 Container Storage Interface (CSI), vos applications Kubernetes peuvent accéder aux objets Amazon S3 via une interface de système de fichiers, ce qui permet d'obtenir un débit agrégé élevé sans modifier le code de l'application.

Cette procédure explique comment déployer le pilote Mountpoint for Amazon S3 CSI Amazon EKS. Avant de poursuivre, veuillez consulter les considérations.

Prérequis

- Un fournisseur IAM (AWS Identity and Access Management) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".
- Version 2.12.3 ou ultérieure de la AWS CLI installée et configurée sur votre appareil ou. AWS CloudShell
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

Étape 1 : créer une politique IAM

Le pilote CSI Mountpoint pour Amazon S3 nécessite des autorisations Amazon S3 pour interagir avec votre système de fichiers. Cette section montre comment créer une politique IAM qui accorde les autorisations nécessaires.

L'exemple de politique suivant suit les recommandations d'autorisation IAM pour Mountpoint. Vous pouvez également utiliser le lien de politique AWS gérée :iam/home?#/policies/arn:aws: iam : :aws:policy/AmazonS3 FullAccess $JSONEditor [AmazonS3, type="console "]FullAccess, mais cette politique gérée accorde plus d'autorisations que ce qui est nécessaire pour Mountpoint.

Pour plus d'informations sur les autorisations recommandées pour Mountpoint, consultez la section Autorisations Mountpoint IAM sur. GitHub

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Politiques.

3. Sur la page Politiques, choisissez Créer une politique.
4. Dans Éditeur de politique, choisissez JSON.
5. Sous Éditeur de politique, copiez et collez ce qui suit :

> **Important**
>
> Remplacez `amzn-s3-demo-bucket1` par votre propre nom de compartiment Amazon S3.

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws: s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws: s3:::amzn-s3-demo-bucket1/*"
            ]
        }
   ]
}
```

Les compartiments de répertoire, introduits avec la classe de stockage Amazon S3 Express One Zone, utilisent un mécanisme d'authentification différent de celui des compartiments à usage général. Au lieu d'utiliser `s3:*` des actions, vous devez utiliser l'`s3express:CreateSession`action. Pour plus d'informations sur les compartiments de répertoire, consultez la section Buckets de répertoire dans le guide de l'utilisateur Amazon S3.

Vous trouverez ci-dessous un exemple de politique de moindre privilège que vous utiliseriez pour un bucket de répertoire.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws: s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket1--usw2-az1--x-s3"
        }
    ]
}
```

6. Choisissez Suivant.
7. Sur la page Vérifier et créer, donnez un nom à votre politique. Cet exemple de démonstration utilise le nom `AmazonS3CSIDriverPolicy`.
8. Choisissez Create Policy (Créer une politique).

Étape 2 : Créer un rôle IAM

Le pilote CSI Mountpoint pour Amazon S3 nécessite des autorisations Amazon S3 pour interagir avec votre système de fichiers. Cette section montre comment créer un rôle IAM pour déléguer ces autorisations. Pour créer ce rôle, vous pouvez utiliser l'un des outils suivants :

- the section called “eksctl”
- the section called “AWS Management Console”
- the section called “AWS CLI”

> **Note**
>
> La politique IAM `AmazonS3CSIDriverPolicy` a été créée dans la section précédente.

eksctl

Pour créer votre rôle IAM de pilote CSI Mountpoint pour Amazon S3 avec **`eksctl`**

Pour créer le rôle IAM et le compte de service Kubernetes, exécutez les commandes suivantes. Ces commandes associent également la politique `AmazonS3CSIDriverPolicy` IAM au rôle, annotent le compte de service Kubernetes (`s3-csi-controller-sa`) avec le nom de ressource Amazon (ARN) du rôle IAM et ajoutent le nom du compte de service Kubernetes à la politique de confiance du rôle IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

AWS Management Console

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionnez une entité de confiance), procédez comme suit :
   a. Dans la section Trusted entity type (Type d'entité de confiance), sélectionnez Web identity (Identité web).
   b. Pour Fournisseur d'identité, choisissez l'URL du fournisseur OpenID Connect pour votre cluster (comme indiqué sous Présentation dans Amazon EKS).

Si ce n' URLs est pas le cas, passez en revue les conditions préalables.

c. Pour Audience, choisissez `sts.amazonaws.com`.

d. Choisissez Suivant.

5. Sur la page Add permissions (Ajouter des autorisations), procédez comme suit :

   a. Dans le champ Politiques de filtrage, saisissez la politique d'AmazonS3CSIDriver.

   > **Note**
   >
   > Cette politique a été créée dans la section précédente.

   b. Cochez la case à gauche du résultat `AmazonS3CSIDriverPolicy` renvoyé par la recherche.

   c. Choisissez Suivant.

6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   a. Dans Nom du rôle, entrez un nom unique pour votre rôle, tel que DriverRole AmazoneKS_S3_CSI_.

   b. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

   c. Choisissez Créer un rôle.

7. Une fois le rôle créé, choisissez le rôle dans la console pour l'ouvrir et le modifier.

8. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la politique d'approbation).

9. Trouvez la ligne qui ressemble à ce qui suit :

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

   Ajoutez une virgule à la fin de la ligne précédente, puis ajoutez la ligne suivante. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. *`EXAMPLED539D4633E53DE1B71EXAMPLE`*Remplacez-le par l'ID du fournisseur OIDC de votre cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10. Assurez-vous que l'`Condition`opérateur est réglé sur`"StringEquals"`.
11. Sélectionnez Update Trust Policy (Mettre à jour la politique d'approbation) pour terminer.

AWS CLI

1. Affichez l'URL du fournisseur OIDC de votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster. Si la sortie de la commande est `None`, consultez Prérequis.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

L'exemple qui suit illustre un résultat.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Créez le rôle IAM, en accordant au compte de service Kubernetes l'action `AssumeRoleWithWebIdentity`.
   a. Copiez le contenu suivant dans un fichier nommé `aws-s3-csi-driver-trust-policy.json`. Remplacez *`111122223333`* par votre ID de compte. Remplacez *`EXAMPLED539D4633E53DE1B71EXAMPLE`* et *`region-code`* par les valeurs renvoyées par l'étape précédente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws: iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
```

```
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:s3-csi-
driver-sa",
          "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. Créez le rôle. Vous pouvez changer le nom de *AmazonEKS_S3_CSI_DriverRole*. Dans ce cas, veillez à le changer également dans les étapes suivantes.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. Attachez la politique IAM créée précédemment au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> La politique IAM `AmazonS3CSIDriverPolicy` a été créée dans la section précédente.

4. Ignorez cette étape si vous installez le pilote en tant que module complémentaire Amazon EKS. Pour les installations autogérées du pilote, créez des comptes de service Kubernetes annotés avec l'ARN du rôle IAM que vous avez créé.

   a. Enregistrez le contenu suivant dans un fichier nommé `mountpoint-s3-service-account.yaml`. Remplacez *111122223333* par votre ID de compte.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
```

```
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

b. Créez le compte de service Kubernetes sur votre cluster. Le compte de service Kubernetes (`mountpoint-s3-csi-controller-sa`) est annoté avec le nom du rôle IAM que vous avez créé. *`AmazonEKS_S3_CSI_DriverRole`*

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> Lorsque vous déployez le plugin dans cette procédure, il crée et est configuré pour utiliser un compte de service nommé `s3-csi-driver-sa`.

Étape 3 : Installation du pilote CSI Mountpoint pour Amazon S3

Vous pouvez installer le pilote CSI Mountpoint pour Amazon S3 via le module complémentaire Amazon EKS. Vous pouvez utiliser les outils suivants pour ajouter le module complémentaire à votre cluster :

- the section called "eksctl"
- the section called "AWS Management Console"
- the section called "AWS CLI"

Vous pouvez également installer le pilote CSI Mountpoint pour Amazon S3 en tant qu'installation autogérée. Pour savoir comment procéder à une installation autogérée, consultez Installation sur GitHub.

À partir `dev1.8.0`, vous pouvez configurer les taches pour tolérer les pods du pilote CSI. Pour ce faire, spécifiez un ensemble personnalisé de teintures à tolérer `node.tolerations` ou tolérez toutes les taches. `node.tolerateAllTaints` Pour plus d'informations, consultez Taints and Tolerations dans la documentation de Kubernetes.

eksctl

Pour ajouter le module complémentaire Amazon S3 CSI à l'aide de **`eksctl`**

Exécutez la commande suivante. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par l'ID de votre compte et *AmazonEKS_S3_CSI_DriverRole* par le nom du rôle IAM créé précédemment.

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole --force
```

Si vous supprimez l'option *--force* et si des paramètres du module complémentaire Amazon EKS entrent en conflit avec vos paramètres existants, alors la mise à jour du module complémentaire Amazon EKS échoue et vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option. Pour plus d'informations sur les autres options de ce paramètre, consultez Modules complémentaires dans la documentation `eksctl`. Pour plus d'informations sur la gestion des champs Amazon EKS Kubernetes, consultez. the section called "Champs que vous pouvez personnaliser"

Vous pouvez le personnaliser `eksctl` à l'aide de fichiers de configuration. Pour plus d'informations, consultez la section Utilisation des valeurs de configuration dans la `eksctl` documentation. L'exemple suivant montre comment tolérer toutes les souillures.

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Mountpoint for Amazon S3 CSI.
4. Choisissez l'onglet Modules complémentaires.

5. Choisissez Obtenez plus de modules complémentaires.
6. Sur la page Sélectionner des modules complémentaires, procédez comme suit :
   a. Dans la section Amazon EKS-Addons, cochez la case Mountpoint for Amazon S3 CSI Driver.
   b. Choisissez Suivant.
7. Sur la page Configurer les paramètres des modules complémentaires sélectionnés, procédez comme suit :
   a. Sélectionnez la version que vous souhaitez utiliser.
   b. Pour Select IAM role, sélectionnez le nom d'un rôle IAM auquel vous avez associé la politique IAM du pilote CSI Mountpoint for Amazon S3.
   c. (Facultatif) Mettez à jour la méthode de résolution des conflits après avoir étendu les paramètres de configuration facultatifs. Si vous sélectionnez Remplacer, un ou plusieurs paramètres du module complémentaire existant peuvent être remplacés par les paramètres du module complémentaire Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur qui en résulte pour résoudre le conflit. Avant de sélectionner cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer vous-même.
   d. (Facultatif) Configurez les tolérances dans le champ Valeurs de configuration après avoir étendu les paramètres de configuration facultatifs.
   e. Choisissez Suivant.
8. Sur la page Vérifier et ajouter, choisissez Créer. Une fois l'installation du module complémentaire terminée, vous pouvez voir le module complémentaire installé.

AWS CLI

Pour ajouter le module complémentaire CSI Mountpoint pour Amazon S3 à l'aide de la CLI AWS

Exécutez la commande suivante. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par l'ID de votre compte, et *AmazonEKS_S3_CSI_DriverRole* par le nom du rôle créé précédemment.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

Vous pouvez personnaliser la commande avec le `--configuration-values` drapeau. L'exemple alternatif suivant montre comment tolérer toutes les souillures.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole \
  --configuration-values '{"node":{"tolerateAllTaints":true}}'
```

Étape 4 : Configuration de Mountpoint pour Amazon S3

Dans la plupart des cas, vous pouvez configurer Mountpoint pour Amazon S3 avec uniquement un nom de compartiment. Pour obtenir des instructions sur la configuration de Mountpoint pour Amazon S3, consultez Configuration de Mountpoint pour Amazon S3 sur. GitHub

Étape 5 : Déployer un exemple d'application

Vous pouvez déployer l'allocation statique au pilote sur un compartiment Amazon S3 existant. Pour plus d'informations, consultez la section Provisionnement statique activé GitHub.

Supprimer le module complémentaire Amazon EKS de Mountpoint pour Amazon S3

Deux options s'offrent à vous pour supprimer le pilote CSI Mountpoint pour Amazon S3.

- Conserver le logiciel complémentaire sur votre cluster : cette option supprime la gestion de tous les paramètres par Amazon EKS. Elle supprime également la possibilité pour Amazon EKS de vous informer des mises à jour et de mettre automatiquement à jour le module complémentaire Amazon EKS après le lancement d'une mise à jour. Toutefois, elle préserve le logiciel complémentaire sur votre cluster. Cette option fait du module complémentaire une installation autogérée plutôt qu'un module complémentaire Amazon EKS. Avec cette option, il n'y a aucun temps d'arrêt pour le module complémentaire. Les commandes dans cette procédure utilisent cette option.
- Supprimer entièrement le logiciel complémentaire de votre cluster : nous vous recommandons de supprimer le module complémentaire Amazon EKS de votre cluster uniquement si aucune ressource de votre cluster n'en dépend. Pour effectuer cette option, supprimez `--preserve` à partir de la commande que vous utilisez dans cette procédure.

Si un compte IAM est associé au module complémentaire, celui-ci n'est pas supprimé.

Vous pouvez utiliser les outils suivants pour supprimer le module complémentaire Amazon S3 CSI :

- the section called "eksctl"
- the section called "AWS Management Console"
- the section called "AWS CLI"

eksctl

Pour supprimer le module complémentaire Amazon S3 CSI à l'aide de **`eksctl`**

Remplacez *`my-cluster`* avec le nom de votre cluster, puis exécutez la commande suivante.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez supprimer le module complémentaire CSI Amazon EBS.
4. Choisissez l'onglet Modules complémentaires.
5. Choisissez Mountpoint pour Amazon S3 CSI Driver.
6. Sélectionnez Remove (Supprimer).
7. Dans la boîte de dialogue de confirmation Remove : aws-mountpoint-s 3-csi-driver, procédez comme suit :
   a. Si vous voulez qu'Amazon EKS cesse de gérer les paramètres du module complémentaire, sélectionnez Conserver sur le cluster. Faites ceci si vous voulez retenir le logiciel du module complémentaire sur votre cluster. Ceci afin que vous puissiez gérer vous-même tous les paramètres du module complémentaire.
   b. Saisissez `aws-mountpoint-s3-csi-driver`.
   c. Sélectionnez Remove (Retirer).

AWS CLI

Pour supprimer le module complémentaire Amazon S3 CSI à l'aide de la CLI AWS

Remplacez *my-cluster* avec le nom de votre cluster, puis exécutez la commande suivante.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver --preserve
```

Activer la fonctionnalité de capture instantanée pour les volumes CSI

La fonctionnalité de capture instantanée permet de point-in-time faire des copies de vos données. Pour que cette fonctionnalité fonctionne dans Kubernetes, vous avez besoin à la fois d'un pilote CSI prenant en charge les snapshots (tel que le pilote Amazon EBS CSI) et d'un contrôleur de snapshots CSI. Le Snapshot Controller est disponible en tant que module complémentaire géré par Amazon EKS ou en tant qu'installation autogérée.

Voici quelques éléments à prendre en compte lors de l'utilisation du contrôleur d'instantané CSI.

- Le contrôleur de capture instantanée doit être installé à côté d'un pilote CSI doté de la fonctionnalité de capture instantanée. Pour les instructions d'installation du pilote Amazon EBS CSI, consultez. the section called "Amazon EBS"
- Kubernetes ne prend pas en charge les instantanés de volumes servis via une migration CSI, tels que les volumes Amazon EBS utilisant un fournisseur with. `StorageClass` `kubernetes.io/aws-ebs` Les volumes doivent être créés avec un `StorageClass` qui fait référence au provisionneur de pilote CSI, `ebs.csi.aws.com`.
- Le mode automatique Amazon EKS n'inclut pas le contrôleur de capture instantanée. La capacité de stockage du mode automatique EKS est compatible avec le contrôleur de capture instantanée.

Nous vous recommandons d'installer le contrôleur d'instantané CSI via le module complémentaire géré Amazon EKS. Ce module complémentaire inclut les définitions de ressources personnalisées (CRDs) nécessaires pour créer et gérer des instantanés sur Amazon EKS. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called "Création d'un module complémentaire". Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

Sinon, si vous souhaitez une installation autogérée du contrôleur de capture instantanée CSI, consultez la section Utilisation dans le système `external-snapshotter` Kubernetes en amont sur. GitHub

Configuration de la mise en réseau pour les clusters Amazon EKS

Votre cluster Amazon EKS est créé dans un VPC. La mise en réseau des pods est fournie par le plug-in Amazon VPC Container Network Interface (CNI) pour les nœuds qui s'exécutent sur une infrastructure. AWS Si vous utilisez des nœuds sur votre propre infrastructure, consultezthe section called "Configurer CNI". Ce chapitre inclut les rubriques suivantes pour en savoir plus sur la mise en réseau de votre cluster.

Rubriques

- Ajouter un sous-réseau VPC existant à un cluster Amazon EKS depuis la console de gestion
- Afficher les exigences réseau d'Amazon EKS pour les VPC et les sous-réseaux
- Créez un Amazon VPC pour votre cluster Amazon EKS
- Afficher les exigences relatives aux groupes de sécurité Amazon EKS pour les clusters
- Gérez les modules complémentaires réseau pour les clusters Amazon EKS

Ajouter un sous-réseau VPC existant à un cluster Amazon EKS depuis la console de gestion

1. Accédez à votre cluster dans la console de gestion
2. Dans l'onglet Mise en réseau, sélectionnez Gérer les ressources VPC
3. Dans la liste déroulante Sous-réseaux, sélectionnez des sous-réseaux supplémentaires dans le VPC de votre cluster.

Pour créer un nouveau sous-réseau VPC :

- Passez en revue les exigences relatives au sous-réseau EKS
- Consultez la section Créer un sous-réseau dans le guide de l'utilisateur d'Amazon Virtual Private Cloud.

Afficher les exigences réseau d'Amazon EKS pour les VPC et les sous-réseaux

Lorsque vous créez un cluster, vous spécifiez un VPC et au moins deux sous-réseaux situés dans des zones de disponibilité différentes. Cette rubrique fournit une vue d'ensemble des exigences et considérations spécifiques à Amazon EKS qui sont requises pour le VPC et les sous-réseaux que vous utilisez avec votre cluster. Si vous n'avez pas de VPC à utiliser avec Amazon EKS, consultez. the section called "Création d'un VPC" Si vous créez un cluster local ou étendu sur AWS Outposts, consultez the section called "Créer un VPC et des sous-réseaux" plutôt cette rubrique. Le contenu de cette rubrique s'applique aux clusters Amazon EKS dotés de nœuds hybrides. Pour connaître les exigences de mise en réseau supplémentaires pour les nœuds hybrides, consultezthe section called "Préparer la mise en réseau".

Exigences et considérations requises pour le VPC

Lorsque vous créez un cluster, le VPC que vous spécifiez doit répondre aux exigences et aux considérations suivantes :

- Le VPC doit disposer d'un nombre suffisant d'adresses IP disponibles pour le cluster, pour tous les nœuds et pour les autres ressources Kubernetes que vous souhaitez créer. Si le VPC que vous souhaitez utiliser ne possède pas un nombre suffisant d'adresses IP, essayez d'augmenter le nombre d'adresses IP disponibles.

  Pour ce faire, vous pouvez mettre à jour la configuration du cluster afin de modifier les sous-réseaux et les groupes de sécurité utilisés par le cluster. Vous pouvez effectuer la AWS Management Console mise à jour à partir de la dernière version de la AWS CLI et de `eksctl` la version `v0.164.0-rc.0` ou ultérieure. AWS CloudFormation Vous pouvez avoir besoin de procéder ainsi pour fournir aux sous-réseaux davantage d'adresses IP disponibles afin de réussir la mise à niveau d'une version de cluster.

  > **Important**
  >
  > Tous les sous-réseaux que vous ajoutez doivent appartenir au même ensemble AZs que celui fourni à l'origine lors de la création du cluster. Les nouveaux sous-réseaux doivent répondre à toutes les autres exigences, par exemple disposer d'un nombre suffisant d'adresses IP.
  > Par exemple, supposons que vous avez créé un cluster et spécifié quatre sous-réseaux. Dans l'ordre où vous les avez spécifiés, le premier sous-réseau se trouve dans la zone

de disponibilité `us-west-2a`, les deuxième et troisième sous-réseaux se trouvent dans la zone de disponibilité `us-west-2b` et le quatrième sous-réseau se trouve dans la zone de disponibilité `us-west-2c`. Si vous souhaitez modifier les sous-réseaux, vous devez fournir au moins un sous-réseau dans chacune des trois zones de disponibilité, et les sous-réseaux doivent se trouver dans le même VPC que les sous-réseaux d'origine.

Si vous avez besoin de plus d'adresses IP que les blocs CIDR du VPC, vous pouvez ajouter des blocs CIDR supplémentaires en associant des blocs CIDR (Routage inter-domaines sans classe) supplémentaires à votre VPC. Vous pouvez associer des blocs d'adresses CIDR privés (RFC 1918) et publics (non-RFC 1918) à votre VPC avant ou après la création de votre cluster.

Vous pouvez ajouter des nœuds qui utilisent le nouveau bloc CIDR immédiatement après l'avoir ajouté. Toutefois, étant donné que le plan de contrôle ne reconnaît le nouveau bloc d'adresse CIDR qu'une fois la réconciliation terminée, un cluster peut mettre jusqu'à une heure pour qu'un bloc d'adresse CIDR que vous avez associé à un VPC soit reconnu. Vous pouvez ensuite exécuter les `kubectl port-forward` commandes `kubectl cp`kubectl `exec`,`kubectl logs`, et (ces commandes utilisent le`kubelet API`) pour les nœuds et les pods du nouveau bloc CIDR. De plus, si vous avez des pods qui fonctionnent comme un backend webhook, vous devez attendre que la réconciliation du plan de contrôle soit terminée.

- Évitez les chevauchements de plages d'adresses IP lorsque vous connectez votre cluster EKS à d'autres VPCs via Transit Gateway, le peering VPC ou d'autres configurations réseau. Des conflits d'adresse CIDR se produisent lorsque le CIDR de service de votre cluster EKS chevauche le CIDR d'un VPC connecté. Dans ces scénarios, les adresses ServiceIP ont la priorité sur les ressources VPCs associées à la même adresse IP, bien que le routage du trafic puisse devenir imprévisible et que les applications puissent ne pas se connecter aux ressources prévues.

  Pour éviter les conflits d'adresses CIDR, assurez-vous que le CIDR de votre service EKS ne se chevauche pas avec un VPC connecté CIDRs et conservez un enregistrement centralisé de toutes les assignations d'adresse CIDR. Si vous rencontrez des chevauchements d'adresses CIDR, vous pouvez utiliser une passerelle de transit avec un VPC de services partagés. Pour plus d'informations, consultez Isolated VPCs with shared services et Modèles de conservation des adresses IP routables Amazon EKS VPC dans un réseau hybride. Reportez-vous également à la VPCs section Communication entre les deux parties de la page Considérations relatives aux VPC et aux sous-réseaux du Guide des meilleures pratiques d'EKS.
- Si vous souhaitez que Kubernetes attribue des `IPv6` adresses aux pods et aux services, associez un bloc `IPv6` CIDR à votre VPC. Pour plus d'informations, consultez Associer un bloc IPv6 CIDR à

votre VPC dans le guide de l'utilisateur Amazon VPC. Vous ne pouvez pas utiliser d'`IPv6`adresses avec des pods et des services exécutés sur des nœuds hybrides et vous ne pouvez pas utiliser de nœuds hybrides avec des clusters configurés avec la famille d'adresses `IPv6` IP.

- Le VPC doit prend en charge le nom d'hôte `DNS` et la résolution `DNS`. Dans le cas contraire, les nœuds ne pourront pas s'enregistrer dans votre cluster. Pour plus d'informations, consultez Attributs DNS pour votre VPC dans le guide de l'utilisateur d'Amazon VPC.
- Le VPC peut nécessiter que les points de terminaison du VPC utilisent. AWS PrivateLink Pour de plus amples informations, veuillez consulter the section called "Exigences et considérations requises pour les sous-réseaux".

Si vous avez créé un cluster avec Kubernetes `1.14` ou une version antérieure, Amazon EKS a ajouté l'identification suivante à votre VPC :

Clé	Valeur
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

Cette identification n'a été utilisée que par Amazon EKS. Vous pouvez supprimer l'identification sans affecter vos services. Il n'est pas utilisé avec les clusters de version `1.15` ou ultérieure.

Exigences et considérations requises pour les sous-réseaux

Lorsque vous créez un cluster, Amazon EKS doit créer entre 2 et 4 interfaces réseau Elastic dans les sous-réseaux que vous spécifiez. Ces interfaces réseau permettent la communication entre votre cluster et votre VPC. Ces interfaces réseau activent également des fonctionnalités Kubernetes telles que `kubectl exec` et `kubectl logs`. Chaque interface réseau créée par Amazon EKS possède le texte `Amazon EKS` *`cluster-name`* dans sa description.

Amazon EKS peut créer ses interfaces réseau dans n'importe quel sous-réseau que vous spécifiez lorsque vous créez un cluster. Vous pouvez modifier les sous-réseaux dans lesquels Amazon EKS crée ses interfaces réseau après la création de votre cluster. Lorsque vous mettez à jour la version Kubernetes d'un cluster, Amazon EKS supprime les interfaces réseau d'origine qu'il a créées et crée de nouvelles interfaces réseau. Ces interfaces réseau peuvent être créées dans les mêmes sous-réseaux que les interfaces réseau d'origine ou dans des sous-réseaux différents des interfaces réseau d'origine. Pour contrôler les sous-réseaux dans lesquels les interfaces réseau sont créées,

vous pouvez limiter le nombre de sous-réseaux que vous spécifiez à seulement deux lorsque vous créez un cluster ou mettre à jour les sous-réseaux après la création du cluster.

Exigences requises pour les sous-réseaux des clusters

Les sous-réseaux que vous spécifiez lors de la création ou de la mise à jour d'un cluster doivent répondre aux exigences suivantes :

- Les sous-réseaux doivent comporter chacun au moins six adresses IP à utiliser par Amazon EKS. Toutefois, nous vous recommandons au moins 16 adresses IP.
- Les sous-réseaux doivent se trouver dans au moins deux zones de disponibilité différentes.
- Les sous-réseaux ne peuvent pas résider dans AWS Outposts ou AWS Wavelength. Toutefois, si vous les avez dans votre VPC, vous pouvez déployer des nœuds autogérés et des ressources Kubernetes sur ces types de sous-réseaux. Pour plus d'informations sur les nœuds autogérés, consultezthe section called "Nœuds autogérés".
- Les sous-réseaux peuvent être publics ou privés. Toutefois, nous vous recommandons de spécifier des sous-réseaux privés, si possible. Un sous-réseau public est un sous-réseau avec une table de routage qui inclut une route vers une passerelle Internet, tandis qu'un sous-réseau privé est un sous-réseau avec une table de routage qui n'inclut pas de route vers une passerelle Internet.
- Les sous-réseaux ne peuvent pas résider dans les zones de disponibilité suivantes :

AWS Région	Nom de la région	Zone de disponibilité non autorisée IDs
`us-east-1`	US East (N. Virginia)	`use1-az3`
`us-west-1`	USA Ouest (Californie du Nord)	`usw1-az2`
`ca-central-1`	Canada (Centre)	`cac1-az3`

Utilisation de la famille d'adresses IP par composant

Le tableau suivant indique la famille d'adresses IP utilisée par chaque composant d'Amazon EKS. Vous pouvez utiliser un système de traduction d'adresses réseau (NAT) ou un autre système de compatibilité pour vous connecter à ces composants à partir d'adresses IP sources appartenant à des familles avec la valeur « Non » pour une entrée de table.

Les fonctionnalités peuvent varier en fonction du paramètre IP family (`ipFamily`) du cluster. Ce paramètre modifie le type d'adresses IP utilisé pour le bloc CIDR attribué par Kubernetes aux services. Un cluster avec la valeur de réglage de IPv4 est appelé IPv4 cluster, et un cluster avec la valeur de réglage de IPv6 est appelé IPv6 cluster.

Composant	IPv4 adresses	IPv6 adresses	Adresses à double pile
Point de terminaison public d'API EKS	Oui [1,3]	Oui [1,3]	Oui [1,3]
Point de terminaison VPC de l'API EKS	Oui	Non	Non
Point de terminaison public de l'API EKS Auth (EKS Pod Identity)	Oui[1]	Oui[1]	Oui[1]
Point de terminaison VPC de l'API EKS Auth (EKS Pod Identity)	Oui[1]	Oui[1]	Oui[1]
`IPv4` Point de terminaison public 2 du cluster Kubernetes	Oui	Non	Non
`IPv4` Point de terminaison privé du cluster Kubernetes 2	Oui	Non	Non
`IPv6` Point de terminaison public 2 du cluster Kubernetes	Oui [1,4]	Oui [1,4]	Oui[4]
`IPv6` Point de terminaison privé du cluster Kubernetes 2	Oui [1,4]	Oui [1,4]	Oui[4]

Composant	IPv4 adresses	IPv6 adresses	Adresses à double pile
Sous-réseaux du cluster Kubernetes	Oui[2]	Non	Oui[2]
Adresses IP principales du nœud	Oui[2]	Non	Oui[2]
Plage d'adresses CIDR du cluster pour les adresses IP de service	Oui[2]	Oui[2]	Non
Adresses IP des pods issues du VPC CNI	Oui[2]	Oui[2]	Non
Émetteur IRSA OIDC URLs	Oui [1,3]	Oui [1,3]	Oui [1,3]

Note

[1] Le point de terminaison est à double pile avec à la fois `IPv4` des `IPv6` adresses et. Vos applications extérieures AWS, vos nœuds pour le cluster et vos pods à l'intérieur du cluster peuvent atteindre ce point de terminaison par l'un `IPv4` ou l'autre des moyens`IPv6`.

[2] Vous choisissez entre un `IPv4` cluster et un `IPv6` cluster dans le paramètre IP family (`ipFamily`) du cluster lorsque vous créez un cluster et cela ne peut pas être modifié. Vous devez plutôt choisir un autre paramètre lorsque vous créez un autre cluster et que vous migrez vos charges de travail.

[3] Le point de terminaison à double pile a été introduit en août 2024. Pour utiliser les points de terminaison à double pile avec la AWS CLI, consultez la configuration des points de terminaison à double pile et FIPS dans le AWS SDKs guide de référence des outils et. Voici la liste des nouveaux points de terminaison :

Point de terminaison public d'API EKS

```
eks.region.api.aws
```

Émetteur IRSA OIDC URLs

```
oidc-eks.region.api.aws
```

[4] Le point de terminaison du cluster à double pile a été introduit en octobre 2024. EKS crée le point de terminaison suivant pour les nouveaux clusters créés après cette date et sélectionnés `IPv6` dans le paramètre de famille d'adresses IP (IPFamily) du cluster :

public/private Point final du cluster EKS

```
eks-cluster.region.api.aws
```

Exigences requises pour les sous-réseaux des nœuds

Vous pouvez déployer des nœuds et des ressources Kubernetes sur les mêmes sous-réseaux que ceux que vous avez spécifiés lors de la création de votre cluster. Cependant, cela n'est pas nécessaire. En effet, vous pouvez également déployer des nœuds et des ressources Kubernetes sur des sous-réseaux que vous n'avez pas spécifiés lors de la création du cluster. Si vous déployez des nœuds sur différents sous-réseaux, Amazon EKS ne crée pas d'interfaces réseau de clusters dans ces sous-réseaux. Tout sous-réseau sur lequel vous déployez des nœuds et des ressources Kubernetes doit répondre aux exigences suivantes :

- Les sous-réseaux doivent disposer de suffisamment d'adresses IP disponibles pour déployer tous vos nœuds et ressources Kubernetes.
- Si vous souhaitez que Kubernetes attribue des `IPv6` adresses aux pods et aux services, vous devez avoir un bloc `IPv6` CIDR et un bloc `IPv4` CIDR associés à votre sous-réseau. Pour plus d'informations, consultez Associer un bloc IPv6 CIDR à votre sous-réseau dans le guide de l'utilisateur Amazon VPC. Les tables de routage associées aux sous-réseaux doivent inclure des routes vers les adresses `IPv4` et `IPv6`. Pour plus d'informations, veuillez consulter Routes dans le guide de l'utilisateur d'Amazon VPC. Les pods se voient attribuer uniquement une adresse `IPv6`. Toutefois, les interfaces réseau créées par Amazon EKS pour votre cluster et vos nœuds se voient attribuer une adresse `IPv4` et une adresse `IPv6`.
- Si vous avez besoin d'un accès entrant à vos pods depuis Internet, assurez-vous de disposer d'au moins un sous-réseau public avec suffisamment d'adresses IP disponibles pour déployer des équilibreurs de charge et des entrées. Vous pouvez déployer des équilibreurs de charge sur des sous-réseaux publics. Les équilibreurs de charge peuvent équilibrer la charge des pods dans des

sous-réseaux privés ou publics. Nous vous recommandons de déployer vos nœuds sur des sous-réseaux privés, si possible.

- Si vous envisagez de déployer des nœuds sur un sous-réseau public, ce sous-réseau doit attribuer automatiquement des adresses publiques `IPv4` ou des adresses `IPv6`. Si vous déployez des nœuds sur un sous-réseau privé doté d'un bloc d'adresse CIDR `IPv6` associé, le sous-réseau privé doit également attribuer automatiquement des adresses `IPv6`. Si vous avez utilisé le AWS CloudFormation modèle fourni par Amazon EKS pour déployer votre VPC après le 26 mars 2020, ce paramètre est activé. Si vous avez utilisé les modèles pour déployer votre VPC avant cette date ou si vous utilisez votre propre VPC, vous devez activer ce paramètre manuellement. Pour le modèle, voirthe section called "Création d'un VPC". Pour plus d'informations, consultez Modifier l'attribut d' IPv4 adressage public de votre sous-réseau et Modifier l'attribut d' IPv6 adressage de votre sous-réseau dans le guide de l'utilisateur Amazon VPC.
- Si le sous-réseau sur lequel vous déployez un nœud est un sous-réseau privé et que sa table de routage n'inclut pas de route vers un périphérique de traduction d'adresses réseau (NAT) () ou une passerelle de sortie uniquement (`IPv4`)`IPv6`, ajoutez des points de terminaison VPC à votre VPC. AWS PrivateLink Les points de terminaison VPC sont nécessaires pour tous les AWS services avec lesquels vos nœuds et vos pods ont besoin pour communiquer. Les exemples incluent Amazon ECR, Elastic Load Balancing CloudWatch, Amazon, AWS Security Token Service et Amazon Simple Storage Service (Amazon S3). Le point de terminaison doit inclure le sous-réseau dans lequel se trouvent les nœuds. Tous les AWS services ne prennent pas en charge les points de terminaison VPC. Pour plus d'informations, voir Qu'est-ce que c'est AWS PrivateLink ? et AWS des services qui s'intègrent à AWS PrivateLink. Pour obtenir une liste complète des exigences Amazon EKS, consultez the section called "Clusters privés".
- Si vous souhaitez déployer des équilibreurs de charge sur un sous-réseau, le sous-réseau doit comporter l'identification suivante :
  - Sous-réseaux privés

Clé	Valeur
`kubernetes.io/role/internal-elb`	`1`

  - Sous-réseaux publics

Clé	Valeur
`kubernetes.io/role/elb`	`1`

Lors de la création d'un cluster Kubernetes correspondant à une version `1.18` ou à une version antérieure, Amazon EKS a ajouté la balise suivante à tous les sous-réseaux spécifiés.

Clé	Valeur
`kubernetes.io/cluster/` *`my-cluster`*	`shared`

Lorsque vous créez un nouveau cluster Kubernetes maintenant, Amazon EKS n'ajoute pas la balise à vos sous-réseaux. Si la balise se trouvait sur des sous-réseaux utilisés par un cluster dont la version était antérieure à`1.19`, elle n'était pas automatiquement supprimée des sous-réseaux lors de la mise à jour du cluster vers une version plus récente. La version `2.1.1` ou antérieure du AWS Load Balancer Controller nécessite cette balise. Si vous utilisez une version plus récente du Load Balancer Controller, vous pouvez supprimer l'identification sans interrompre vos services. Pour plus d'informations sur le contrôleur, consultezthe section called " AWS Contrôleur Load Balancer".

Si vous avez déployé un VPC à l'aide de l'un `eksctl` des modèles de AWS CloudFormation VPC Amazon EKS, les règles suivantes s'appliquent :

- À partir du 26 mars 2020 : les adresses `IPv4` publiques sont automatiquement attribuées par des sous-réseaux publics aux nouveaux nœuds déployés sur des sous-réseaux publics.
- Avant le 26 mars 2020 — Les `IPv4` adresses publiques ne sont pas automatiquement attribuées par les sous-réseaux publics aux nouveaux nœuds déployés sur des sous-réseaux publics.

Cette modification affecte les nouveaux groupes de nœuds déployés sur des sous-réseaux publics de la manière suivante :

- Groupes de nœuds gérés : si le groupe de nœuds est déployé sur un sous-réseau public le 22 avril 2020 ou après cette date, l'attribution automatique des adresses IP publiques doit être activée pour le sous-réseau public. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau.

- Groupes de nœuds autogérés Linux, Windows ou Arm : si le groupe de nœuds est déployé sur un sous-réseau public le 26 mars 2020 ou après cette date, l'attribution automatique des adresses IP publiques doit être activée pour le sous-réseau public. Sinon, les nœuds doivent être lancés avec une adresse IP publique. Pour plus d'informations, consultez Modification de l'attribut d' IPv4 adressage public pour votre sous-réseau ou Affectation d'une IPv4 adresse publique lors du lancement de l'instance.

Exigences et considérations requises pour les sous-réseaux partagés

Vous pouvez utiliser le partage VPC pour partager des sous-réseaux avec d'autres AWS comptes au sein des mêmes Organisations. AWS Vous pouvez créer des clusters Amazon EKS dans des sous-réseaux partagés, en tenant compte des points suivants :

- Le propriétaire du sous-réseau VPC doit partager un sous-réseau avec un compte participant avant que ce compte puisse y créer un cluster Amazon EKS.
- Vous ne pouvez pas lancer de ressources à l'aide du groupe de sécurité par défaut du VPC car celui-ci appartient au propriétaire. De plus, les participants ne peuvent pas lancer de ressources avec des groupes de sécurité détenus par d'autres participants ou par le propriétaire.
- Dans un sous-réseau partagé, le participant et le propriétaire contrôlent séparément les groupes de sécurité au sein de leur compte respectif. Le propriétaire du sous-réseau peut voir ces groupes de sécurité créés par les participants, mais ne peut pas exécuter d'actions sur ceux-ci. Si le propriétaire du sous-réseau souhaite supprimer ou modifier ces groupes de sécurité, le participant qui a créé le groupe de sécurité doit effectuer l'action.
- Si un cluster est créé par un participant, il faut tenir compte des éléments suivants :
  - Le rôle IAM de cluster et les rôles IAM de nœud doivent être créés dans ce compte. Pour plus d'informations, consultez the section called “Rôle IAM du cluster” et the section called “Rôle IAM de nœud”.
  - Tous les nœuds doivent être créés par le même participant, y compris les groupes de nœuds gérés.
- Le propriétaire du VPC partagé ne peut pas afficher, mettre à jour ou supprimer un cluster créé par un participant dans le sous-réseau partagé. Cela s'ajoute aux ressources VPC auxquelles chaque compte a un accès différent. Pour plus d'informations, consultez Responsabilités et autorisations pour les propriétaires et les participants dans le Guide de l'utilisateur Amazon VPC.
- Si vous utilisez la fonctionnalité de mise en réseau personnalisée du plug-in Amazon VPC CNI pour Kubernetes, vous devez utiliser les mappages d'ID de zone de disponibilité répertoriés dans

le compte du propriétaire pour créer chacun d'eux. `ENIConfig` Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée".

Pour plus d'informations sur le partage de sous-réseaux VPC, consultez Partager votre VPC avec d'autres comptes dans le Guide de l'utilisateur Amazon VPC.

Créez un Amazon VPC pour votre cluster Amazon EKS

Vous pouvez utiliser Amazon Virtual Private Cloud (Amazon VPC) pour lancer AWS des ressources dans un réseau virtuel que vous avez défini. Ce réseau virtuel ressemble beaucoup à un réseau traditionnel que vous pourriez exécuter dans votre propre centre de données. Toutefois, il présente l'avantage d'utiliser l'infrastructure évolutive d'Amazon Web Services. Il est recommandé de bien connaître le service Amazon VPC avant de déployer des clusters Amazon EKS de production. Pour de plus amples informations, consultez le Guide de l'utilisateur Amazon VPC.

Un cluster Amazon EKS, des nœuds et des ressources Kubernetes sont déployés sur un VPC. Si vous souhaitez utiliser un VPC existant avec Amazon EKS, ce VPC doit répondre aux exigences décrites dans the section called "Exigences requises pour le VPC et les sous-réseaux". Cette rubrique explique comment créer un VPC répondant aux exigences d'Amazon EKS à l'aide d'un modèle fourni par AWS CloudFormation Amazon EKS. Une fois que vous avez déployé un modèle, vous pouvez consulter les ressources créées par le modèle pour savoir exactement quelles ressources il a créées, ainsi que la configuration de ces ressources. Si vous utilisez des nœuds hybrides, votre VPC doit comporter des routes dans sa table de routage pour votre réseau sur site. Pour plus d'informations sur la configuration réseau requise pour les nœuds hybrides, consultezthe section called "Préparer la mise en réseau".

Prérequis

Pour créer un VPC pour Amazon EKS, vous devez disposer des autorisations IAM nécessaires pour créer des ressources Amazon VPC. Ces ressources sont les sous-réseaux VPCs, les groupes de sécurité, les tables de routage et les routes, ainsi que les passerelles Internet et NAT. Pour plus d'informations, consultez la section Créer un VPC avec un sous-réseau public comme exemple de politique dans le guide de l'utilisateur Amazon VPC et la liste complète des actions dans la référence d'autorisation de service.

Vous pouvez créer un VPC avec des sous-réseaux à la fois publics et privés, uniquement avec des sous-réseaux publics ou uniquement avec des sous-réseaux privés.

Sous-réseaux publics et privés

Ce VPC comporte deux sous-réseaux publics et deux privés. La table de routage associée à un sous-réseau public possède une route vers une passerelle Internet. Cependant, la table de routage d'un sous-réseau privé ne possède pas de route vers une passerelle Internet. Un sous-réseau public et un sous-réseau privé sont déployés dans la même zone de disponibilité. Les autres sous-réseaux publics et privés sont déployés dans une deuxième zone de disponibilité de la même AWS région. Nous recommandons cette option pour la plupart des déploiements.

Avec cette option, vous pouvez déployer vos nœuds sur des sous-réseaux privés. Cette option permet à Kubernetes de déployer des équilibreurs de charge sur les sous-réseaux publics afin d'équilibrer la charge du trafic vers les pods exécutés sur les nœuds des sous-réseaux privés. Les `IPv4` adresses publiques sont automatiquement attribuées aux nœuds déployés sur des sous-réseaux publics, mais aucune `IPv4` adresse publique n'est attribuée aux nœuds déployés sur des sous-réseaux privés.

Vous pouvez également attribuer des adresses `IPv6` aux nœuds des sous-réseaux publics et privés. Les nœuds des sous-réseaux privés peuvent communiquer avec le cluster et d'autres AWS services. Les pods peuvent communiquer avec Internet via une passerelle NAT à l'aide d'`IPv4`adresses ou une passerelle Internet sortante uniquement à l'aide d'`IPv6`adresses déployées dans chaque zone de disponibilité. Un groupe de sécurité est déployé et comporte des règles qui refusent tout trafic entrant provenant de sources autres que le cluster ou les nœuds mais autorise tout le trafic sortant. Les sous-réseaux sont balisés de sorte que Kubernetes puisse y déployer des équilibreurs de charge.

a. Ouvrez la AWS CloudFormation console.

b. Dans la barre de navigation, sélectionnez une AWS région compatible avec Amazon EKS.

c. Sélectionnez Créer une pile, Avec de nouvelles ressources (standard).

d. Sous Prerequisite - Prepare template (Prérequis - Préparer le modèle), assurez-vous que Template is ready (Le modèle est prêt) est sélectionné, puis sous Spécifier un modèle, sélectionnez Amazon S3 URL (URL Amazon S3).

e. Vous pouvez créer un VPC qui prend en charge uniquement `IPv4`, ou un VPC qui prend en charge `IPv4` et `IPv6`. Collez l'un des éléments suivants URLs dans la zone de texte sous l'URL Amazon S3 et choisissez Next :

   - `IPv4`

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

- IPv4 et IPv6

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

a. Dans la page Spécifier les détails de la pile, entrez les paramètres, puis choisissez Suivant.

- Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez utiliser le nom du modèle que vous avez utilisé à l'étape précédente. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
- VpcBlock: Choisissez une plage d'IPv4adresses CIDR pour votre VPC. Une IPv4 adresse provenant de ce bloc est attribuée à chaque nœud, pod et équilibreur de charge que vous déployez. Les IPv4 valeurs par défaut fournissent suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez les modifier. Pour plus d'informations, consultez Dimensionnement des VPC et des sous-réseaux dans le Guide de l'utilisateur Amazon VPC. Vous pouvez également ajouter des blocs CIDR supplémentaires au VPC une fois celui-ci créé. Si vous créez un IPv6 VPC, des plages IPv6 CIDR vous sont automatiquement attribuées à partir de l'espace d'adresses Global Unicast d'Amazon.
- PublicSubnet01Block : Spécifiez un bloc IPv4 CIDR pour le sous-réseau public 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un IPv6 VPC, ce bloc vous est spécifié dans le modèle.
- PublicSubnet02Block : Spécifiez un bloc IPv4 CIDR pour le sous-réseau public 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un IPv6 VPC, ce bloc vous est spécifié dans le modèle.
- PrivateSubnet01Block : Spécifiez un bloc IPv4 CIDR pour le sous-réseau privé 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un IPv6 VPC, ce bloc vous est spécifié dans le modèle.

- PrivateSubnet02Block : Spécifiez un bloc `IPv4` CIDR pour le sous-réseau privé 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un `IPv6` VPC, ce bloc vous est spécifié dans le modèle.

b. (Facultatif) Sur la page Configure stack options (Configurer les options de pile), étiquetez vos ressources de pile, puis choisissez Next (Suivant).

c. Sur la page Review (Vérification), choisissez Create stack (Créer une pile).

d. Lorsque votre pile est créée, sélectionnez-la dans la console et choisissez Outputs (Sorties).

e. Enregistrez le VpcIdpour le VPC qui a été créé. Vous en avez besoin lors de la création de votre cluster et de vos nœuds.

f. Enregistrez SubnetIdsles sous-réseaux créés et indiquez si vous les avez créés en tant que sous-réseaux publics ou privés. Vous en avez besoin d'au moins deux lors de la création de votre cluster et de vos nœuds.

g. Si vous avez créé un VPC `IPv4`, ignorez cette étape. Si vous avez créé un VPC `IPv6`, vous devez activer l'option d'attribution automatique d'adresses `IPv6` pour les sous-réseaux publics qui ont été créés par le modèle. Ce paramètre est déjà activé pour les sous-réseaux privés. Pour activer le paramètre, effectuez les étapes suivantes :

   i. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

   ii. Dans le volet de navigation de gauche, sélectionnez Subnets (Sous-réseaux).

   iii. Sélectionnez l'un de vos sous-réseaux publics (***stack-name***/SubnetPublic01 ou ***stack-name***/SubnetPublic02 contient le mot public) et choisissez Actions, Modifier les paramètres du sous-réseau.

   iv. Cochez la case Activer l'attribution automatique d' IPv6 adresses, puis sélectionnez Enregistrer.

   v. Effectuez à nouveau les étapes précédentes pour votre autre sous-réseau public.

Sous-réseaux publics uniquement

Ce VPC possède trois sous-réseaux publics déployés dans différentes zones de disponibilité d'une région. AWS Tous les nœuds se voient attribuer automatiquement des adresses `IPv4` publiques et peuvent envoyer et recevoir du trafic Internet via une passerelle Internet. Un groupe de sécurité est déployé qui refuse tout le trafic entrant et autorise tout le trafic sortant. Les sous-réseaux sont balisés de sorte que Kubernetes puisse y déployer des équilibreurs de charge.

a. Ouvrez la AWS CloudFormation console.

b. Dans la barre de navigation, sélectionnez une AWS région compatible avec Amazon EKS.

c. Choisissez Create stack (Créer une pile), Avec de nouvelles ressources (standard).

d. Sous Prepare template (Préparer le modèle), assurez-vous que Prepare template (Modèle est prêt) est sélectionné, puis sous Template source (Source du modèle), sélectionnez Amazon S3 URL (URL Amazon S3).

e. Collez l'URL suivante dans la zone de texte sous URL Amazon S3 et sélectionnez Next (Suivant) :

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

a. Dans la page Spécifier les détails, entrez les paramètres, puis choisissez Suivant.

- Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *`amazon-eks-vpc-sample`*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
- VpcBlock: Choisissez un bloc CIDR pour votre VPC. Une `IPv4` adresse provenant de ce bloc est attribuée à chaque nœud, pod et équilibreur de charge que vous déployez. Les `IPv4` valeurs par défaut fournissent suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez les modifier. Pour plus d'informations, consultez Dimensionnement des VPC et des sous-réseaux dans le Guide de l'utilisateur Amazon VPC. Vous pouvez également ajouter des blocs CIDR supplémentaires au VPC une fois celui-ci créé.
- Subnet01Block : spécifiez un bloc d'adresse CIDR pour le sous-réseau 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
- Subnet02Block : spécifiez un bloc d'adresse CIDR pour le sous-réseau 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
- Subnet03Block : spécifiez un bloc d'adresse CIDR pour le sous-réseau 3. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.

b. (Facultatif) Sur la page Options (Options), étiquetez les ressources de votre pile. Choisissez Next (Suivant).

c. Sur la page Review (Vérification), choisissez Create (Créer).

d. Lorsque votre pile est créée, sélectionnez-la dans la console et choisissez Outputs (Sorties).

e. Enregistrez le VpcIdpour le VPC qui a été créé. Vous en avez besoin lors de la création de votre cluster et de vos nœuds.

f. Enregistrez le SubnetIdspour les sous-réseaux qui ont été créés. Vous en avez besoin d'au moins deux lors de la création de votre cluster et de vos nœuds.

g. (Facultatif) Tout cluster que vous déployez sur ce VPC peut attribuer des `IPv4` adresses privées à vos pods et services. Si vous souhaitez déployer des clusters sur ce VPC pour attribuer des `IPv6` adresses privées à vos pods et services, mettez à jour votre VPC, votre sous-réseau, vos tables de routage et vos groupes de sécurité. Pour plus d'informations, consultez la section Migrer les VPCs fichiers existants de IPv4 vers IPv6 dans le guide de l'utilisateur Amazon VPC. Amazon EKS requiert que l'option `Auto-assign` d'adresses `IPv6` soit activée pour vos sous-réseaux. Par défaut, elle est désactivée.

Sous-réseaux privés uniquement

Ce VPC possède trois sous-réseaux privés déployés dans différentes zones de disponibilité de la région. AWS Les ressources déployées sur les sous-réseaux ne peuvent pas accéder à Internet, pas plus qu'Internet ne peut accéder aux ressources des sous-réseaux. Le modèle crée des points de terminaison VPC à l'aide AWS PrivateLink de plusieurs AWS services auxquels les nœuds ont généralement besoin d'accéder. Si vos nœuds ont besoin d'un accès Internet sortant, vous pouvez ajouter une passerelle NAT publique dans la zone de disponibilité de chaque sous-réseau après la création du VPC. Un groupe de sécurité est créé qui refuse tout trafic entrant, à l'exception des ressources déployées dans les sous-réseaux. Un groupe de sécurité autorise également tout le trafic sortant. Les sous-réseaux sont balisés de sorte que Kubernetes puisse y déployer des équilibreurs de charge. Si vous créez un VPC avec cette configuration, reportez-vous à la section the section called "Clusters privés" pour connaître les exigences et considérations supplémentaires.

a. Ouvrez la AWS CloudFormation console.

b. Dans la barre de navigation, sélectionnez une AWS région compatible avec Amazon EKS.

c. Choisissez Create stack (Créer une pile), Avec de nouvelles ressources (standard).

d. Sous Prepare template (Préparer le modèle), assurez-vous que Prepare template (Modèle est prêt) est sélectionné, puis sous Template source (Source du modèle), sélectionnez Amazon S3 URL (URL Amazon S3).

e. Collez l'URL suivante dans la zone de texte sous URL Amazon S3 et sélectionnez Next (Suivant) :

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-
fully-private-vpc.yaml
```

a. Sur la page Spécifier les détails, renseignez les paramètres, puis choisissez Suivant.
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *amazon-eks-fully-private-vpc*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
   - VpcBlock: Choisissez un bloc CIDR pour votre VPC. Une `IPv4` adresse provenant de ce bloc est attribuée à chaque nœud, pod et équilibreur de charge que vous déployez. Les `IPv4` valeurs par défaut fournissent suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez les modifier. Pour plus d'informations, consultez Dimensionnement des VPC et des sous-réseaux dans le Guide de l'utilisateur Amazon VPC. Vous pouvez également ajouter des blocs CIDR supplémentaires au VPC une fois celui-ci créé.
   - PrivateSubnet01Block : Spécifiez un bloc CIDR pour le sous-réseau 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
   - PrivateSubnet02Block : Spécifiez un bloc CIDR pour le sous-réseau 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
   - PrivateSubnet03Block : Spécifiez un bloc CIDR pour le sous-réseau 3. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.

b. (Facultatif) Sur la page Options (Options), étiquetez les ressources de votre pile. Choisissez Next (Suivant).

c. Sur la page Review (Vérification), choisissez Create (Créer).

d. Lorsque votre pile est créée, sélectionnez-la dans la console et choisissez Outputs (Sorties).

e. Enregistrez le VpcIdpour le VPC qui a été créé. Vous en avez besoin lors de la création de votre cluster et de vos nœuds.

f. Enregistrez le SubnetIdspour les sous-réseaux qui ont été créés. Vous en avez besoin d'au moins deux lors de la création de votre cluster et de vos nœuds.

g. (Facultatif) Tout cluster que vous déployez sur ce VPC peut attribuer des `IPv4` adresses privées à vos pods et services. Si vous souhaitez déployer des clusters sur ce VPC pour attribuer des

IPv6 adresses privées à vos pods et services, mettez à jour votre VPC, votre sous-réseau, vos tables de routage et vos groupes de sécurité. Pour plus d'informations, consultez la section Migrer les VPCs fichiers existants de IPv4 vers IPv6 dans le guide de l'utilisateur Amazon VPC. Amazon EKS exige que l'option `Auto-assign IPv6` adresses soit activée sur vos sous-réseaux (elle est désactivée par défaut).

Afficher les exigences relatives aux groupes de sécurité Amazon EKS pour les clusters

Cette rubrique décrit les exigences de groupe de sécurité d'un cluster Amazon EKS.

Groupe de sécurité du cluster par défaut

Lorsque vous créez un cluster, Amazon EKS crée un groupe de sécurité nomméeks-cluster-sg-*my-cluster-uniqueID* . Ce groupe de sécurité dispose des règles par défaut suivantes :

Type de règle	Protocole	Ports	Source	Destination
Entrant	Tous	Tous	Auto-utilisateur	
Sortant	Tous	Tous		0.0.0.0/0 (IPv4) ou : :/0 () IPv6
Sortant	Tous	Tous		Self (pour le trafic EFA)

Le groupe de sécurité par défaut inclut une règle sortante qui autorise le trafic Elastic Fabric Adapter (EFA) ayant pour destination le même groupe de sécurité. Cela permet d'activer le trafic EFA au sein du cluster, ce qui est bénéfique pour les charges de travail AI/ML et HPC (High Performance Computing). Pour plus d'informations, consultez l'adaptateur Elastic Fabric pour les charges de travail AI/ML et HPC sur Amazon EC2 dans le guide de l'utilisateur d'Amazon Elastic Compute Cloud.

> **Important**
>
> Si votre cluster n'a pas besoin de la règle sortante, vous pouvez la supprimer. Si vous la supprimez, vous devez toujours avoir les règles minimales énumérées dans la section

Restriction du trafic du cluster. Si vous supprimez la règle entrante, Amazon EKS la recrée à chaque fois que le cluster est mis à jour.

Amazon EKS ajoute les balises suivantes au groupe de sécurité. Si vous supprimez les balises, Amazon EKS les ajoute à nouveau au groupe de sécurité à chaque fois que le cluster est mis à jour.

Clé	Valeur
`kubernetes.io/cluster/ my-cluster`	`owned`
`aws:eks:cluster-name`	*my-cluster*
`Name`	`eks-cluster-sg-` *my-cluster -uniqueid*

Amazon EKS associe automatiquement ce groupe de sécurité aux ressources suivantes qu'il crée également :

- 2 à 4 interfaces réseau élastiques (désignées pour le reste de ce document sous le nom d'interface réseau) créés lors de la création de votre cluster.
- Interfaces réseau des nœuds dans n'importe quel groupe de nœuds géré que vous créez.

Les règles par défaut permettent à tout le trafic de circuler librement entre votre cluster et vos nœuds, et autorise tout le trafic sortant vers n'importe quelle destination. Lorsque vous créez un cluster, spécifiez (éventuellement) vos propres groupes de sécurité. Si c'est le cas, Amazon EKS associe également les groupes de sécurité que vous spécifiez aux interfaces réseau qu'il crée pour votre cluster. Toutefois, il ne les associe à aucun groupe de nœuds que vous créez.

Vous pouvez déterminer l'ID du groupe de sécurité de votre cluster dans la AWS Management Console section Mise en réseau du cluster. Vous pouvez également le faire en exécutant la commande AWS CLI suivante.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Restriction du trafic de cluster

Si vous devez limiter les ports ouverts entre le cluster et les nœuds, vous pouvez supprimer la règle sortante par défaut et ajouter les règles minimales suivantes qui sont requises pour le cluster. Si vous supprimez la règle entrante par défaut, Amazon EKS la recrée à chaque fois que le cluster est mis à jour.

Type de règle	Protocole	Port	Destination
Sortant	TCP	443	Groupe de sécurité du cluster
Sortant	TCP	10250	Groupe de sécurité du cluster
Sortant (DNS)	TCP et UDP	53	Groupe de sécurité du cluster

Vous devez également ajouter des règles pour le trafic suivant :

- Tout protocole et port que vos nœuds peuvent utiliser pour la communication entre les nœuds.
- Accès Internet sortant permettant aux nœuds d'accéder à Amazon EKS APIs pour l'introspection du cluster et l'enregistrement des nœuds au moment du lancement. Si vos nœuds n'ont pas accès à Internet, consultez Déployer des clusters privés avec un accès Internet limité pour des considérations supplémentaires.
- Accès aux nœuds pour extraire des images de conteneurs depuis Amazon ECR ou d'autres registres de conteneurs APIs dont ils ont besoin pour extraire des images, tels que. DockerHub Pour plus d'informations, consultez la section Plages d'adresses AWS IP dans le manuel de référence AWS général.
- Accès au nœud Amazon S3.
- Des règles distinctes sont requises pour les adresses `IPv4` et `IPv6`.
- Si vous utilisez des nœuds hybrides, vous devez ajouter un groupe de sécurité supplémentaire à votre cluster pour permettre la communication avec vos nœuds et pods locaux. Pour de plus amples informations, veuillez consulter the section called “Préparer la mise en réseau”.

Si vous envisagez de limiter les règles, nous vous recommandons de tester minutieusement tous vos pods avant d'appliquer les règles modifiées à un cluster de production.

Si vous avez initialement déployé un cluster avec Kubernetes `1.14` et une version plateforme `eks.3` ou antérieure, prenez en considération les éléments suivants :

- Vous pouvez également disposer de groupes de sécurité de nœud et de plan de contrôle. Lorsque ces groupes ont été créés, ils incluaient les règles restreintes répertoriées dans le tableau précédent. Ces groupes de sécurité ne sont plus nécessaires et peuvent être supprimés. Toutefois, vous devez vous assurer que le groupe de sécurité de votre cluster contient les règles que ces groupes contiennent.
- Si vous avez déployé le cluster directement à l'aide de l'API ou si vous avez utilisé un outil tel que la AWS CLI ou AWS CloudFormation pour créer le cluster et que vous n'avez pas spécifié de groupe de sécurité lors de la création du cluster, le groupe de sécurité par défaut pour le VPC a été appliqué aux interfaces réseau du cluster créées par Amazon EKS.

Groupes de sécurité partagés

Amazon EKS prend en charge les groupes de sécurité partagés.

- Les associations VPC de groupe de sécurité associent des groupes de sécurité à plusieurs VPCs groupes d'un même compte et d'une même région.
  - Découvrez comment associer des groupes de sécurité à plusieurs VPCs dans le guide de l'utilisateur Amazon VPC.
- Les groupes de sécurité partagés vous permettent de partager des groupes de sécurité avec d'autres AWS comptes. Les comptes doivent appartenir à la même AWS organisation.
  - Découvrez comment partager des groupes de sécurité avec des organisations dans le guide de l'utilisateur Amazon VPC.
- Les groupes de sécurité sont toujours limités à une seule AWS région.

Considérations relatives à Amazon EKS

- EKS a les mêmes exigences pour les groupes de sécurité partagés ou multi-VPC que pour les groupes de sécurité standard.

Gérez les modules complémentaires réseau pour les clusters Amazon EKS

Plusieurs modules complémentaires de mise en réseau sont disponibles pour votre cluster Amazon EKS.

Modules complémentaires intégrés

> **Note**
>
> Lorsque vous créez un cluster EKS :
>
> - Utilisation de la AWS console : les modules complémentaires intégrés (tels que CoreDNS, kube-proxy, etc.) sont automatiquement installés en tant que modules complémentaires Amazon EKS. Ils peuvent être facilement configurés et mis à jour via la AWS console, la CLI ou SDKs.
> - Utilisation d'autres méthodes (CLI SDKs, etc.) : les mêmes modules complémentaires intégrés sont installés sous forme de versions autogérées qui s'exécutent comme des déploiements Kubernetes réguliers. Ils nécessitent une configuration et des mises à jour manuelles car ils ne peuvent pas être gérés par le biais AWS d'outils.
>
> Nous recommandons d'utiliser des modules complémentaires Amazon EKS plutôt que des versions autogérées afin de simplifier la gestion des modules complémentaires et de permettre une configuration et des mises à jour centralisées par le biais de AWS services.

Plug-in Amazon VPC CNI pour Kubernetes

Ce module complémentaire CNI crée des interfaces réseau élastiques et les attache à vos EC2 nœuds Amazon. Le module complémentaire attribue également une adresse privée `IPv4` ou une `IPv6` adresse provenant de votre VPC à chaque pod et service. Par défaut, ce module complémentaire est installé sur votre cluster. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC". Si vous utilisez des nœuds hybrides, le VPC CNI est toujours installé par défaut, mais il ne peut pas s'exécuter sur vos nœuds hybrides en raison d'une règle anti-affinité. Pour plus d'informations sur vos options CNI pour les nœuds hybrides, consultezthe section called "Configurer CNI".

CoreDNS

CoreDNS est un serveur DNS flexible et extensible qui peut servir de DNS de cluster Kubernetes. CoreDNS fournit une résolution de nom pour tous les pods du cluster. Par défaut, ce module complémentaire est installé sur votre cluster. Pour de plus amples informations, veuillez consulter the section called "CoreDNS".

`kube-proxy`

Ce module complémentaire gère les règles du réseau sur vos EC2 nœuds Amazon et permet la communication réseau avec vos Pods. Par défaut, ce module complémentaire est installé sur votre cluster. Pour de plus amples informations, veuillez consulter the section called "kube-proxy".

Compléments AWS réseau optionnels

AWS Contrôleur Load Balancer

Lorsque vous déployez des objets de service Kubernetes de type Kubernetes`loadbalancer`, le contrôleur crée des équilibreurs de charge AWS réseau. Lorsque vous créez des objets d'entrée Kubernetes, le contrôleur crée AWS des équilibreurs de charge d'application. Nous recommandons d'utiliser ce contrôleur pour provisionner des équilibreurs de charge réseau, plutôt que d'utiliser l'ancien contrôleur Cloud Provider intégré à Kubernetes. Pour plus d'informations, consultez la documentation du contrôleur AWS Load Balancer Controller.

AWS Contrôleur d'API Gateway

Ce contrôleur vous permet de connecter des services sur plusieurs clusters Kubernetes à l'aide de l'API de passerelle Kubernetes. Le contrôleur connecte les services Kubernetes exécutés sur des EC2 instances Amazon, des conteneurs et des fonctions sans serveur à l'aide du service Amazon VPC Lattice. Pour plus d'informations, veuillez consulter la documentation Contrôleur d'API Gateway AWS.

Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

Attribuer IPs à des pods avec l'Amazon VPC CNI

> **Tip**
>
> Avec le mode automatique d'Amazon EKS, vous n'avez pas besoin d'installer ou de mettre à niveau des modules complémentaires réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Le plug-in Amazon VPC CNI pour Kubernetes est déployé sur chaque nœud EC2 Amazon de votre cluster Amazon EKS. Le module complémentaire crée des interfaces réseau élastiques et les attache à vos EC2 nœuds Amazon. Le module complémentaire attribue également une adresse privée `IPv4` ou une `IPv6` adresse provenant de votre VPC à chaque pod.

Une version du module complémentaire est déployée avec chaque nœud Fargate de votre cluster, mais vous ne la mettez pas à jour sur les nœuds Fargate. D'autres plug-ins CNI compatibles peuvent être utilisés sur les clusters Amazon EKS, mais il s'agit du seul plug-in CNI pris en charge par Amazon EKS pour les nœuds exécutés sur AWS une infrastructure. Pour plus d'informations sur les autres plug-ins CNI compatibles, consultezthe section called “Plugins CNI alternatifs”. Le VPC CNI n'est pas pris en charge pour une utilisation avec des nœuds hybrides. Pour plus d'informations sur vos options CNI pour les nœuds hybrides, consultezthe section called “Configurer CNI”.

Le tableau suivant répertorie la dernière version disponible du type de module complémentaire Amazon EKS pour chaque version de Kubernetes.

Versions CNI d'Amazon VPC

Version de Kubernetes	Type de version VPC CNI d'Amazon EKS
1,33	v1.20.0-eksbuild.1
1,32	v1.20.0-eksbuild.1
1,31	v1.20.0-eksbuild.1
1,30	v1.20.0-eksbuild.1
1,29	v1.20.0-eksbuild.1

Version de Kubernetes	Type de version VPC CNI d'Amazon EKS
1,28	v1.20.0-eksbuild.1
1,27	v1.20.0-eksbuild.1
1,26	v1.19.6-eksbuild.1

> **Important**
>
> Si vous gérez vous-même ce module complémentaire, les versions du tableau ne sont peut-être pas les mêmes que les versions autogérées disponibles. Pour plus d'informations sur la mise à jour du type autogéré de ce module complémentaire, consultez la rubrique the section called "Mise à jour (autogérée)".

> **Important**
>
> Pour passer à VPC CNI v1.12.0 ou version ultérieure, vous devez d'abord effectuer la mise à niveau vers VPC CNI v1.7.0. Nous vous recommandons de mettre à jour une version mineure à la fois.

Considérations

Voici quelques considérations relatives à l'utilisation de cette fonctionnalité.

- Les versions sont spécifiées comme `major-version.minor-version.patch-version-eksbuild.build-number`.
- Vérifiez la compatibilité des versions pour chaque fonctionnalité. Certaines fonctionnalités de chaque version du plugin Amazon VPC CNI pour Kubernetes nécessitent certaines versions de Kubernetes. Lorsque vous utilisez différentes fonctionnalités d'Amazon EKS, si une version spécifique du module complémentaire est requise, cela est indiqué dans la documentation des fonctionnalités. Nous vous recommandons d'utiliser la dernière version, sauf si vous avez une raison précise d'utiliser une version antérieure.

Création de l'Amazon VPC CNI (module complémentaire Amazon EKS)

Suivez les étapes ci-dessous pour créer le plug-in Amazon VPC CNI pour Kubernetes, le module complémentaire Amazon EKS.

Avant de commencer, passez en revue les considérations. Pour de plus amples informations, veuillez consulter the section called "Considérations".

Prérequis

Les conditions suivantes sont requises pour le plug-in Amazon VPC CNI pour Kubernetes, le module complémentaire Amazon EKS.

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route.
- Un fournisseur IAM (AWS Identity and Access Management) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".
- Un rôle IAM auquel est attachée la politique IAM Amazoneks_CNI_Policy (si votre cluster utilise la famille) ou une politique (si votre cluster utilise `IPv4` la famille). IPv6 `IPv6` Pour plus d'informations sur le rôle VPC CNI, consultez. the section called "Configuration pour IRSA" Pour plus d'informations sur IPv6 cette politique, consultezthe section called "Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`".

> **Important**
>
> Plug-in Amazon VPC CNI pour les versions de Kubernetes afin d'implémenter la version `v1.16.0` de spécification CNI. `v1.16.1` `v1.0.0` Pour plus d'informations sur `v1.0.0` la spécification CNI, voir Spécification de l'interface réseau de conteneurs (CNI) sur. GitHub

Procédure

Après avoir rempli les conditions requises, suivez les étapes ci-dessous pour créer le module complémentaire.

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.16.4-eksbuild.2
```

2. Déterminez le type de module complémentaire installé sur votre cluster. Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

   Si un numéro de version est renvoyé, le type Amazon EKS du module complémentaire est installé sur votre cluster et vous n'avez pas besoin de suivre les étapes restantes de cette procédure. Si une erreur est renvoyée, le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Suivez les étapes restantes de cette procédure pour l'installer.

3. Enregistrez la configuration du module complémentaire actuellement installé.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Créez le module complémentaire à l'aide de la AWS CLI. Si vous souhaitez utiliser le AWS Management Console ou `eksctl` pour créer le module complémentaire, consultez the section called "Création d'un module complémentaire" et spécifiez `vpc-cni` le nom du module complémentaire. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.

   - Remplacez *my-cluster* par le nom de votre cluster.
   - *v1.20.0-eksbuild.1*Remplacez-le par la dernière version répertoriée dans le tableau des dernières versions de votre cluster. Pour le tableau des dernières versions, voirthe section called "Versions CNI d'Amazon VPC".
   - *111122223333*Remplacez-le par votre identifiant de compte et *AmazonEKSVPCCNIRole* par le nom d'un rôle IAM existant que vous avez créé. Pour spécifier un rôle, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.0-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
```

Si vous avez appliqué des paramètres personnalisés à votre module complémentaire actuel qui entrent en conflit avec les paramètres par défaut du module complémentaire Amazon EKS, la création risque d'échouer. Si la création échoue, vous recevez un message d'erreur qui peut vous aider à résoudre le problème. Vous pouvez également ajouter `--resolve-conflicts OVERWRITE` à la commande précédente. Cela permet au module complémentaire de remplacer les paramètres personnalisés existants. Une fois que vous avez créé le module complémentaire, vous pouvez le mettre à jour avec vos paramètres personnalisés.

5. Vérifiez que la dernière version du module complémentaire pour la version Kubernetes de votre cluster a été ajoutée à votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

La création du module complémentaire peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
v1.20.0-eksbuild.1
```

6. Si vous avez défini des paramètres personnalisés pour votre module complémentaire d'origine, avant de créer le module complémentaire Amazon EKS, utilisez la configuration que vous avez enregistrée lors d'une étape précédente pour mettre à jour le module complémentaire EKS avec vos paramètres personnalisés. Suivez les étapes de the section called "Mise à jour (module complémentaire EKS)".
7. (Facultatif) Installez le `cni-metrics-helper` sur votre cluster. Il collecte les informations relatives à l'interface Elastic Network et aux adresses IP, les agrège au niveau du cluster et publie les statistiques sur Amazon. CloudWatch Pour plus d'informations, voir cni-metrics-helperci-dessous GitHub.

Mettre à jour le Amazon VPC CNI (module complémentaire Amazon EKS)

Mettez à jour le type Amazon EKS du module complémentaire Amazon VPC CNI pour Kubernetes. Si vous n'avez pas ajouté le type Amazon EKS du module complémentaire à votre cluster, vous pouvez l'installer en suivant les instructions ci-dessousthe section called "Création". Vous pouvez également mettre à jour l'autre type d'installation VPC CNI en suivant. the section called "Mise à jour (autogérée)"

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

L'exemple qui suit illustre un résultat.

```
v1.20.0-eksbuild.1
```

Comparez la version avec le tableau des dernières versions surthe section called "Versions CNI d'Amazon VPC". Si la version renvoyée est identique à la version de Kubernetes de votre cluster indiquée dans le tableau des dernières versions, cela signifie que la dernière version est déjà installée sur votre cluster et vous n'avez pas besoin de suivre le reste de cette procédure. Si vous recevez une erreur, au lieu d'un numéro de version dans votre sortie, cela signifie que le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Vous devez créer le module complémentaire avant de pouvoir le mettre à jour à l'aide de cette procédure. Pour créer le type Amazon EKS du module complémentaire VPC CNI, vous pouvez suivre. the section called "Création"

2. Enregistrez la configuration du module complémentaire actuellement installé.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. Mettez à jour votre module complémentaire à l'aide de la AWS CLI. Si vous souhaitez utiliser AWS Management Console ou mettre `eksctl` à jour le module complémentaire, consultezthe section called "Mettre à jour un module complémentaire". Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.

   - Remplacez *my-cluster* par le nom de votre cluster.
   - *v1.20.0-eksbuild.1*Remplacez-le par la dernière version répertoriée dans le tableau des dernières versions de votre cluster.
   - *111122223333*Remplacez-le par votre identifiant de compte et *AmazonEKSVPCCNIRole* par le nom d'un rôle IAM existant que vous avez créé. Pour créer un rôle IAM pour le VPC CNI, consultez. the section called "Étape 1 : créer le plugin Amazon VPC CNI pour le rôle IAM de Kubernetes" Pour spécifier un rôle, vous devez disposer d'un fournisseur IAM OpenID Connect

(OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

- L'`--resolve-conflicts PRESERVE`option préserve les valeurs de configuration existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres complémentaires et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `OVERWRITE`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour des paramètres, ils peuvent être remplacés par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par`none`, Amazon EKS ne modifie pas la valeur des paramètres, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.
- Si vous ne mettez pas à jour un paramètre de configuration, `--configuration-values '{"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'` supprimez-le de la commande. Si vous mettez à jour un paramètre de configuration, `"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}` remplacez-le par le paramètre que vous souhaitez définir. Dans cet exemple, la variable d'environnement `AWS_VPC_K8S_CNI_EXTERNALSNAT` est définie sur `true`. La valeur que vous spécifiez doit être valide pour le schéma de configuration. Si vous ne connaissez pas le schéma de configuration`aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.20.0-eksbuild.1` , exécutez-le en le `v1.20.0-eksbuild.1` remplaçant par le numéro de version du module complémentaire dont vous souhaitez consulter la configuration. Le schéma est renvoyé dans la sortie. Si vous avez une configuration personnalisée existante, que vous souhaitez tout supprimer et rétablir les valeurs de tous les paramètres sur les valeurs par défaut d'Amazon EKS, supprimez-la `"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}` de la commande afin d'avoir le champ vide`{}`. Pour une explication de chaque paramètre, voir Variables de configuration CNI activées GitHub.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.0-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

La mise à jour peut prendre plusieurs secondes.

4. Assurez-vous que la version du module complémentaire a été mise à jour. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

La mise à jour peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.20.0-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws: eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn": "arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

Mettre à jour le Amazon VPC CNI (module complémentaire autogéré)

> **Important**
>
> Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si vous ne connaissez pas la différence entre les types, consultez the section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS

à votre cluster, consultez the section called "Création d'un module complémentaire". Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

1. Vérifiez que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

   Si un message d'erreur est renvoyé, le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Pour gérer vous-même le module complémentaire, suivez les étapes restantes de cette procédure afin de le mettre à jour. Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Pour le mettre à jour, suivez la procédure décrite dans la rubrique the section called "Mettre à jour un module complémentaire" plutôt que cette procédure. Si vous ne connaissez pas les différences entre les types de modules complémentaires, consultezthe section called "Modules complémentaires Amazon EKS".

2. Découvrez quelle version de l'image de conteneur est actuellement installée sur votre cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

   L'exemple qui suit illustre un résultat.

```
v1.20.0-eksbuild.1
```

   Il est possible que votre sortie n'inclue pas le numéro de build.

3. Backup vos paramètres actuels afin de pouvoir configurer les mêmes paramètres une fois que vous aurez mis à jour votre version.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

   Pour consulter les versions disponibles et vous familiariser avec les modifications apportées à la version vers laquelle vous souhaitez effectuer la mise à jour, consultez les versions publiées sur GitHub. Notez que nous vous recommandons de procéder à la même mise à jour`major`.

`minor`. `patch`version répertoriée dans le tableau des dernières versions disponibles, même si des versions ultérieures sont disponibles sur GitHub. Pour le tableau des dernières versions disponibles, voirthe section called "Versions CNI d'Amazon VPC". Les versions de build répertoriées dans le tableau ne sont pas spécifiées dans les versions autogérées répertoriées sur GitHub. Mettez à jour votre version en procédant comme suit, selon le cas :

- Si vous n'avez aucun paramètre personnalisé pour le module complémentaire, exécutez la commande sous l'`To apply this release:`en-tête correspondant à la version vers laquelle vous souhaitez effectuer la mise à jour. GitHub
- Si vous avez personnalisé des paramètres, téléchargez le fichier manifeste avec la commande suivante. Modifiez *https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/master/aws-k8s-cni.yaml* l'URL de la version vers GitHub laquelle vous souhaitez effectuer la mise à jour.

  ```
  curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/
  master/aws-k8s-cni.yaml
  ```

  Si nécessaire, modifiez le manifeste avec les paramètres personnalisés de la sauvegarde que vous avez effectuée lors d'une étape précédente, puis appliquez le manifeste modifié à votre cluster. Si vos nœuds n'ont pas accès aux référentiels Amazon EKS Amazon ECR privés d'où proviennent les images (voir les lignes commençant par le manifeste), vous devrez télécharger les images, les copier `image:` dans votre propre référentiel et modifier le manifeste pour extraire les images de votre référentiel. Pour de plus amples informations, veuillez consulter the section called "Copier une image dans un référentiel".

  ```
  kubectl apply -f aws-k8s-cni.yaml
  ```

4. Vérifiez que la nouvelle version est maintenant installée sur votre cluster.

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.20.0
   ```

5. (Facultatif) Installez le `cni-metrics-helper` sur votre cluster. Il collecte les informations relatives à l'interface Elastic Network et aux adresses IP, les agrège au niveau du cluster et publie

les statistiques sur Amazon. CloudWatch Pour plus d'informations, voir cni-metrics-helperci-dessous GitHub.

Configurer le plug-in Amazon VPC CNI pour utiliser IRSA

Le plug-in Amazon VPC CNI pour Kubernetes est le plug-in réseau pour la mise en réseau des Pod dans les clusters Amazon EKS. Le plugin est chargé d'attribuer les adresses IP VPC aux pods Kubernetes et de configurer le réseau nécessaire pour les pods sur chaque nœud.

> **Note**
>
> Le plug-in Amazon VPC CNI prend également en charge Amazon EKS Pod Identities. Pour de plus amples informations, veuillez consulter the section called "Attribuer un rôle IAM".

Le plugin :

- Nécessite des autorisations AWS Identity and Access Management (IAM). Si votre cluster utilise cette `IPv4` famille, les autorisations sont spécifiées dans la politique `AmazonEKS_CNI_Policy` AWS gérée. Si votre cluster utilise cette `IPv6` famille, les autorisations doivent être ajoutées à une politique IAM que vous créez ; pour obtenir des instructions, reportez-vous the section called "Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`" à. Vous pouvez attacher cette politique au rôle IAM de nœud Amazon EKS ou à un rôle IAM distinct. Pour obtenir des instructions sur la manière d'associer la politique au rôle IAM du nœud Amazon EKS, consultezthe section called "Rôle IAM de nœud". Nous vous recommandons de l'affecter à un rôle séparé, comme détaillé dans cette rubrique.
- Crée et est configuré pour utiliser un compte de service Kubernetes nommé `aws-node` lors de son déploiement. Le compte de service est lié à un `clusterrole` Kubernetes nommé `aws-node`, qui reçoit les autorisations Kubernetes requises.

> **Note**
>
> Les pods du plugin Amazon VPC CNI pour Kubernetes ont accès aux autorisations attribuées au rôle IAM du nœud Amazon EKS, sauf si vous bloquez l'accès à IMDS. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

- Nécessite un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route.
- Nécessite un fournisseur de gestion des AWS identités et des accès (IAM) (IAM) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Étape 1 : créer le plugin Amazon VPC CNI pour le rôle IAM de Kubernetes

1. Déterminez la famille d'adresses IP de votre cluster.

```
aws eks describe-cluster --name my-cluster | grep ipFamily
```

L'exemple qui suit illustre un résultat.

```
"ipFamily": "ipv4"
```

La sortie peut renvoyer `ipv6` à la place.

2. Créez le rôle IAM. Vous pouvez utiliser `eksctl` ou `kubectl` et la AWS CLI pour créer votre rôle IAM.

eksctl

- Créez un rôle IAM et attachez la politique IAM au rôle avec la commande qui correspond à la famille d'adresses IP de votre cluster. La commande crée et déploie une AWS CloudFormation pile qui crée un rôle IAM, y attache la politique que vous spécifiez et annote le compte de `aws-node` service Kubernetes existant avec l'ARN du rôle IAM créé.
  - `IPv4`

    Remplacez *`my-cluster`* par votre propre valeur.

```
eksctl create iamserviceaccount \
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --override-existing-serviceaccounts \
    --approve
```

  - `IPv6`

Remplacez *my-cluster* par votre propre valeur. Remplacez *111122223333* par l'ID de votre compte et remplacez *AmazonEKS_CNI_IPv6_Policy* par le nom de votre politique IPv6. Si vous n'avez pas de IPv6 politique, reportez-vous the section called "Créer une politique IAM pour les clusters qui utilisent la famille IPv6" à la section pour en créer une. Pour utiliser IPv6 avec votre cluster, ce dernier doit répondre à plusieurs exigences. Pour de plus amples informations, veuillez consulter the section called "IPv6".

```
eksctl create iamserviceaccount \
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn arn:aws: iam::111122223333:policy/
AmazonEKS_CNI_IPv6_Policy \
    --override-existing-serviceaccounts \
    --approve
```

kubectl et la CLI AWS

i. Consultez l'URL du fournisseur OIDC de votre cluster.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

L'exemple qui suit illustre un résultat.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

Si aucune sortie n'est renvoyée, vous devez créer un fournisseur OIDC IAM pour votre cluster.

ii. Copiez le contenu suivant dans un fichier nommé *vpc-cni-trust-policy.json*. Remplacez *111122223333* par votre ID de compte et *EXAMPLED539D4633E53DE1B71EXAMPLE* par la sortie renvoyée à l'étape précédente. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

iii. Créez le rôle. Vous pouvez remplacer *AmazonEKSVPCCNIRole* par n'importe quel nom que vous choisissez.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. Attachez la politique IAM requise au rôle. Exécutez la commande qui correspond à la famille d'adresses IP de votre cluster.

- `IPv4`

  ```
  aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name AmazonEKSVPCCNIRole
  ```

- `IPv6`

  Remplacez *111122223333* par l'ID de votre compte et *AmazonEKS_CNI_IPv6_Policy* par le nom de votre politique `IPv6`. Si vous n'avez pas de `IPv6` politique, reportez-vous the section called "Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`" à la section pour en créer une. Pour utiliser `IPv6` avec votre cluster, ce dernier doit répondre à plusieurs exigences. Pour de plus amples informations, veuillez consulter the section called "IPv6".

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

v. Exécutez la commande suivante pour annoter le compte de service `aws-node` avec l'ARN du rôle IAM que vous avez créé précédemment. Remplacez les *`example values`* par vos propres valeurs.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. (Facultatif) Configurez le type de point de terminaison du service AWS Security Token utilisé par votre compte de service Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Points de terminaison STS".

Étape 2 : redéployer le plugin Amazon VPC CNI pour Kubernetes Pods

1. Supprimez et recréez tous les pods existants associés au compte de service pour appliquer les variables d'environnement d'identification. L'annotation n'est pas appliquée aux pods actuellement en cours d'exécution sans l'annotation. La commande suivante supprime les `aws-node` DaemonSet pods existants et les déploie avec l'annotation du compte de service.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Vérifiez que les Pods ont tous redémarré.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Décrivez l'un des pods et vérifiez que les variables d'`AWS_ROLE_ARN`environnement `AWS_WEB_IDENTITY_TOKEN_FILE` et existent. *`cpjw7`*Remplacez-le par le nom de l'un de vos pods renvoyé dans le résultat de l'étape précédente.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

L'exemple qui suit illustre un résultat.

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:           /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Deux ensembles de résultats dupliqués sont renvoyés car le Pod contient deux conteneurs. Les deux exemples présentent les mêmes valeurs.

Si votre Pod utilise le point de terminaison AWS régional, la ligne suivante est également renvoyée dans la sortie précédente.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Étape 3 : suppression de la politique CNI du rôle IAM de nœud

Si le rôle IAM de votre nœud Amazon EKS est actuellement associé à la politique `AmazonEKS_CNI_Policy` IAM (`IPv4`) ou à une IPv6 politique, et que vous avez créé un rôle IAM distinct, que vous lui avez attaché la politique et que vous l'avez attribué au compte de service `aws-node` Kubernetes, nous vous recommandons de supprimer la politique de votre rôle de nœud à l'aide de la commande AWS CLI correspondant à la famille d'adresses IP de votre cluster. Remplacez *`AmazonEKSNodeRole`* par le nom de votre rôle de nœud.

- `IPv4`

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::aws:policy/AmazonEKS_CNI_Policy
```

- `IPv6`

  Remplacez *`111122223333`* par l'ID de votre compte et *`AmazonEKS_CNI_IPv6_Policy`* par le nom de votre politique `IPv6`.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

Créer une politique IAM pour les clusters qui utilisent la famille **`IPv6`**

Si vous avez créé un cluster qui utilise cette `IPv6` famille et que le cluster possède une version `1.10.1` ou une version ultérieure du module complémentaire Amazon VPC CNI pour Kubernetes configurée, vous devez créer une politique IAM que vous pouvez attribuer à un rôle IAM. Si vous avez un cluster existant que vous n'avez pas configuré avec la `IPv6` famille lorsque vous l'avez créé`IPv6`, vous devez créer un nouveau cluster pour pouvoir l'utiliser. Pour plus d'informations sur l'utilisation de `IPv6` avec votre cluster, veuillez consulter the section called "IPv6".

1. Copiez la politique suivante et enregistrez-la dans un fichier appelé `vpc-cni-ipv6-policy.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. Créez la politique IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

En savoir plus sur les modes et la configuration du VPC CNI

Le plug-in Amazon VPC CNI pour Kubernetes fournit une mise en réseau pour les Pods. Consultez le tableau suivant pour en savoir plus sur les fonctionnalités réseau disponibles.

Fonctionnalité réseau	En savoir plus
Configurez votre cluster pour attribuer IPv6 des adresses aux clusters, aux pods et aux services	the section called "IPv6"
Utiliser IPv4 la traduction d'adresses réseau source pour les pods	the section called "Trafic sortant"
Limitez le trafic réseau à destination et en provenance de vos Pods	the section called "Restreindre le trafic"
Personnalisation de l'interface réseau secondaire dans les nœuds	the section called "Mise en réseau personnalisée"
Augmentez le nombre d'adresses IP de votre nœud	the section called "Augmenter le nombre d'adresses IP"
Utiliser des groupes de sécurité pour le trafic réseau du Pod	the section called "Groupes de sécurité pour Pods"
Utiliser plusieurs interfaces réseau pour les Pods	the section called "Interfaces multiples"

En savoir plus sur IPv6 les adresses des clusters, des pods et des services

S'applique à : Pods avec EC2 instances Amazon et Fargate Pods

Par défaut, Kubernetes attribue des `IPv4` adresses à vos pods et services. Au lieu d'attribuer des `IPv4` adresses à vos pods et services, vous pouvez configurer votre cluster pour qu'il leur attribue `IPv6` des adresses. Amazon EKS ne prend pas en charge les pods ou services à double empilement, même si Kubernetes le fait. Par conséquent, vous ne pouvez pas attribuer à la fois `IPv4` des `IPv6` adresses et des adresses à vos pods et services.

Vous sélectionnez la famille IP que vous souhaitez utiliser pour votre cluster lorsque vous le créez. Vous ne pouvez pas modifier la famille après avoir créé le cluster.

Pour un didacticiel sur le déploiement d'un `IPv6` cluster Amazon EKS, consultezthe section called "Déploiement".

Voici les points à prendre en compte lors de l'utilisation de cette fonctionnalité :

`IPv6`Support des fonctionnalités

- Pas de support Windows : les modules et services Windows ne sont pas pris en charge.
- EC2 Nœuds basés sur Nitro requis : vous ne pouvez les utiliser qu'**`IPv6`**avec des nœuds AWS Amazon ou EC2 Fargate basés sur Nitro.
- EC2 et nœuds Fargate pris en charge : vous pouvez les utiliser **`IPv6`** the section called "Groupes de sécurité pour Pods" avec les nœuds Amazon et les EC2 nœuds Fargate.
- Outposts non pris en charge : vous ne pouvez pas les utiliser `IPv6` avec. Amazon EKS sur AWS Outposts
- FSx car Lustre n'est pas pris en charge : Le n'the section called "Amazon FSx pour Lustre"est pas pris en charge.
- La mise en réseau personnalisée n'est pas prise en charge : si vous avez déjà contribué the section called "Mise en réseau personnalisée" à atténuer l'épuisement des adresses IP, vous pouvez l'utiliser à la `IPv6` place. Vous ne pouvez pas utiliser le réseau personnalisé avec`IPv6`. Si vous utilisez un réseau personnalisé pour l'isolation du réseau, vous devrez peut-être continuer à utiliser un réseau personnalisé et la famille `IPv4` pour vos clusters.

Attributions d'adresses IP

- Services Kubernetes : seules des adresses sont attribuées aux services Kubernetes. `IPv6` Aucune IPv4 adresse ne leur est attribuée.
- Pods : une IPv6 adresse et une adresse locale de l'hôte IPv4 sont attribuées aux pods. L' IPv4 adresse locale de l'hôte est attribuée à l'aide d'un plug-in CNI local lié à un VPC CNI et l'adresse n'est pas signalée au plan de contrôle Kubernetes. Il n'est utilisé que lorsqu'un pod doit communiquer avec une IPv4 ressource externe dans un autre Amazon VPC ou sur Internet. L' IPv4 adresse locale de l'hôte accède SNATed (par VPC CNI) à l' IPv4 adresse principale de l'ENI principal du nœud de travail.
- Pods et services : Les pods et services ne reçoivent que `IPv6` des adresses, pas `IPv4` des adresses. Lorsque les pods doivent communiquer avec des `IPv4` points de terminaison externes,

ils utilisent le NAT sur le nœud lui-même. Cette fonctionnalité NAT intégrée élimine le besoin de DNS64 et NAT64. Pour le trafic nécessitant un accès public à Internet, le trafic du Pod est l'adresse réseau source traduite en adresse IP publique.

- Adresses de routage : lorsqu'un pod communique en dehors du VPC, son `IPv6` adresse d'origine est préservée (elle n'est pas traduite en `IPv6` adresse du nœud). Ce trafic est acheminé directement via une passerelle Internet ou une passerelle Internet de sortie uniquement.
- Nœuds : une `IPv6` adresse `IPv4` et est attribuée à tous les nœuds.
- Pods Fargate : chaque Fargate Pod reçoit `IPv6` une adresse du CIDR spécifiée pour le sous-réseau dans lequel il est déployé. L'unité matérielle sous-jacente qui exécute les Fargate Pods obtient une adresse `IPv6` et une adresse `IPv4` uniques attribuées CIDRs au sous-réseau dans lequel l'unité matérielle est déployée.

Comment utiliser **`IPv6`** avec EKS

- Créer un nouveau cluster : vous devez créer un nouveau cluster et spécifier que vous souhaitez utiliser la `IPv6` famille pour ce cluster. Vous ne pouvez pas activer la `IPv6` famille pour un cluster que vous avez mis à jour à partir d'une version précédente. Pour obtenir des instructions sur la création d'un nouveau cluster, consultez la section Considérations.
- Utiliser une version récente du VPC CNI : Déployez la version d'Amazon VPC CNI ou une version ultérieure. `1.10.1` Cette version ou une version ultérieure est déployée par défaut. Après avoir déployé le module complémentaire, vous ne pouvez pas rétrograder votre module complémentaire Amazon VPC CNI vers une version `1.10.1` inférieure à la version sans avoir préalablement supprimé tous les nœuds de tous les groupes de nœuds de votre cluster.
- Configurer le VPC CNI pour **`IPv6`** : Si vous utilisez des EC2 nœuds Amazon, vous devez configurer le module complémentaire Amazon VPC CNI avec délégation de préfixe IP et. `IPv6` Si vous choisissez la famille `IPv6` lors de la création de votre cluster, la version `1.10.1` du module complémentaire correspond par défaut à cette configuration. C'est le cas pour un module complémentaire auto-géré ou Amazon EKS. Pour de plus amples informations sur la délégation de préfixes IP, veuillez consulter the section called “Augmenter le nombre d'adresses IP”.
- Configuration **`IPv4`** et **`IPv6`** adresses : lorsque vous créez un cluster, le VPC et les sous-réseaux que vous spécifiez doivent avoir un bloc `IPv6` CIDR attribué au VPC et aux sous-réseaux que vous spécifiez. Ils doivent également avoir un bloc d'adresse CIDR `IPv4` qui leur est affecté. En effet, même si vous souhaitez uniquement utiliser `IPv6`, un VPC nécessite toujours un bloc d'adresse CIDR `IPv4` pour fonctionner. Pour plus d'informations, consultez Associer un bloc IPv6 CIDR à votre VPC dans le guide de l'utilisateur Amazon VPC.

- Attribuer automatiquement IPv6 des adresses aux nœuds : Lorsque vous créez vos nœuds, vous devez spécifier des sous-réseaux configurés pour attribuer automatiquement `IPv6` des adresses. Dans le cas contraire, vous ne pouvez pas déployer vos nœuds. Par défaut, cette configuration est désactivée. Pour plus d'informations, consultez Modifier l'attribut d' IPv6 adressage de votre sous-réseau dans le guide de l'utilisateur Amazon VPC.
- Définissez les tables de routage à utiliser **`IPv6`** : les tables de routage attribuées à vos sous-réseaux doivent comporter des itinéraires pour les `IPv6` adresses. Pour de plus amples informations, veuillez consulter la section Migrer vers IPv6 du Guide de l'utilisateur Amazon VPC.
- Définissez des groupes de sécurité pour **`IPv6`** : Vos groupes de sécurité doivent autoriser `IPv6` les adresses. Pour de plus amples informations, veuillez consulter la section Migrer vers IPv6 du Guide de l'utilisateur Amazon VPC.
- Configurer l'équilibreur de charge : utilisez la version `2.3.1` ou une version ultérieure du AWS Load Balancer Controller pour équilibrer la charge des applications HTTP à l'aide the section called “Répartition de la charge des applications” du ou du trafic réseau utilisant les deux pods, the section called “Répartition de charge réseau” l'`IPv6`un des équilibreurs de charge étant en mode IP, mais pas en mode instance. Pour de plus amples informations, veuillez consulter the section called “ AWS Contrôleur Load Balancer”.
- Ajouter une stratégie **`IPv6`** IAM : vous devez associer une stratégie `IPv6` IAM au rôle IAM ou CNI IAM de votre nœud. Entre les deux, nous vous recommandons de le rattacher à un rôle CNI IAM. Pour plus d'informations, consultez the section called “Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`” et the section called “Étape 1 : créer le plugin Amazon VPC CNI pour le rôle IAM de Kubernetes”.
- Évaluez tous les composants : effectuez une évaluation approfondie de vos applications, des modules complémentaires Amazon EKS et AWS des services auxquels vous intégrez avant de déployer `IPv6` des clusters. Cela permet de s'assurer que tout fonctionne comme prévu avec `IPv6`.
- Ajouter des groupes de nœuds **`BootstrapArguments`** autogérés : lors de la création d'un groupe de nœuds autogérés dans un cluster utilisant la `IPv6` famille, les données utilisateur doivent inclure les éléments suivants `BootstrapArguments` pour le fichier bootstrap.sh qui s'exécute au démarrage du nœud. *`your-cidr`*Remplacez-le par la plage `IPv6` CIDR du VPC de votre cluster.

  ```
  --ip-family ipv6 --service-ipv6-cidr your-cidr
  ```

  Si vous ne connaissez pas la `IPv6 CIDR` plage de votre cluster, vous pouvez la voir à l'aide de la commande suivante (nécessite la version de la AWS CLI `2.4.9` ou une version ultérieure).

```
aws eks describe-cluster --name my-cluster --query
 cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
```

Déploiement d'un **IPv6** cluster Amazon EKS et de nœuds Amazon Linux gérés

Dans ce didacticiel, vous allez déployer un IPv6 Amazon VPC, un cluster Amazon EKS appartenant à la IPv6 famille et un groupe de nœuds géré avec des nœuds EC2 Amazon Linux. Vous ne pouvez pas déployer de nœuds Amazon EC2 Windows dans un IPv6 cluster. Vous pouvez également déployer des nœuds Fargate sur votre cluster, bien que ces instructions ne soient pas fournies dans cette rubrique pour des raisons de simplicité.

Prérequis

Avant de commencer le didacticiel, procédez comme suit :

Installez et configurez les outils et ressources suivants dont vous avez besoin pour créer et gérer un cluster Amazon EKS.

- Nous vous recommandons de vous familiariser avec tous les paramètres et de déployer un cluster avec les paramètres qui répondent à vos besoins. Pour plus d'informations, consultez the section called "Créer un cluster "the section called "Groupes de nœuds gérés", et les considérations relatives à cette rubrique. Vous ne pouvez activer certains paramètres que lors de la création de votre cluster.
- L'outil de ligne de commande kubectl est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est 1.29, vous pouvez utiliser la version kubectl 1.28, 1.29 ou 1.30. Pour installer ou mettre à niveau kubectl, veuillez consulter the section called "Configurer kubectl et eksctl".
- Le principal de sécurité IAM que vous utilisez doit être autorisé à utiliser les rôles IAM Amazon EKS, les rôles liés à un service AWS CloudFormation, un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et utilisation de rôles liés à un service dans le guide de l'utilisateur IAM.
- Si vous utilisez eksctl, installez la version 0.210.0 ou ultérieure sur votre ordinateur. Pour l'installer ou le mettre à jour, veuillez consulter Installation dans la documentation de eksctl.
- Version 2.12.3 ou version ultérieure 1.27.160 ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour

vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur. Si vous utilisez le AWS CloudShell, vous devrez peut-être installer la version 2.12.3 ou ultérieure ou 1.27.160 ou ultérieure de la CLI, car la version de la AWSAWS CLI par défaut installée dans le AWS CloudShell peut être une version antérieure.

Vous pouvez utiliser eksctl ou CLI pour déployer un `IPv6` cluster.

Déployer un IPv6 cluster avec eksctl

a. Créez le fichier `ipv6-cluster.yaml`. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

   - Remplacez *`my-cluster`* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Remplacez *`region-code`* par n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général.
   - La valeur de `version` correspond à la version de votre cluster. Pour plus d'informations, consultez le eks/latest/userguide/kubernetes fichier -versions.html [Versions prises en charge par Amazon EKS, type="documentation "].
   - Remplacer *`my-nodegroup`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.
   - Remplacez *`t3.medium`* par tout type d'instance du système AWS Nitro.

```
cat >ipv6-cluster.yaml <<EOF
---
```

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

b. Créer votre cluster.

```
eksctl create cluster -f ipv6-cluster.yaml
```

La création du cluster prend plusieurs minutes. Ne poursuivez pas avant de voir la dernière ligne de sortie, qui ressemble à la sortie suivante.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

c. Vérifiez que des `IPv6` adresses sont attribuées aux pods par défaut.

```
kubectl get pods -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME                        READY   STATUS    RESTARTS   AGE     IP
                     NODE                                            NOMINATED NODE
 READINESS GATES
aws-node-rslts              1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh              1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2    1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8    1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk            1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g            1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

d. Vérifiez que les adresses `IPv6` sont attribuées aux services par défaut.

```
kubectl get services -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

e. (Facultatif) Déployez un exemple d'application ou déployez le AWS Load Balancer Controller et un exemple d'application pour équilibrer la charge des applications HTTP the section called "Répartition de la charge des applications" ou le trafic réseau avec the section called "Répartition de charge réseau" les `IPv6` pods.

f. Une fois que vous avez terminé d'utiliser le cluster et les nœuds que vous avez créés pour ce didacticiel, vous devez nettoyer les ressources que vous avez créées à l'aide de la commande suivante.

```
eksctl delete cluster my-cluster
```

Déployer un IPv6 cluster avec une AWS CLI

> **Important**
>
> - Vous devez effectuer toutes les étapes de cette procédure avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :
>
> ```
> aws sts get-caller-identity
> ```
>
> - Vous devez effectuer toutes les étapes de cette procédure dans le même shell. Plusieurs étapes utilisent des variables définies dans les étapes précédentes. Les étapes qui utilisent des variables ne fonctionneront pas correctement si les valeurs des variables sont définies dans un shell différent. Si vous utilisez le AWS CloudShellpour effectuer la procédure suivante, n'oubliez pas que si vous n'interagissez pas avec le clavier ou le pointeur pendant environ 20 à 30 minutes, votre session shell se termine. Les processus en cours d'exécution ne sont pas considérés comme des interactions.
> - Les instructions sont écrites pour le shell Bash et peuvent nécessiter des ajustements dans d'autres shells.

Remplacez toutes les *example values* dans les étapes de cette procédure par vos propres valeurs.

a. Exécutez les commandes suivantes pour définir certaines variables utilisées dans les étapes suivantes. Remplacez *region-code* par la AWS région dans laquelle vous souhaitez déployer vos ressources. La valeur peut être n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général. Remplacez *my-cluster* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacer *my-nodegroup* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une

lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Remplacez *111122223333* par votre ID de compte.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

b. Créez un Amazon VPC avec des sous-réseaux publics et privés qui répondent aux exigences Amazon EKS et `IPv6`.

   i. Exécutez la commande suivante pour définir une variable pour le nom de votre AWS CloudFormation pile. Vous pouvez remplacer *my-eks-ipv6-vpc* par n'importe quel nom que vous choisissez.

   ```
   export vpc_stack_name=my-eks-ipv6-vpc
   ```

   ii. Créez un `IPv6` VPC à l'aide d'un AWS CloudFormation modèle.

   ```
   aws cloudformation create-stack --region $region_code --stack-name $vpc_stack_name
    \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
   cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
   ```

   La création de la pile prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante tant que le résultat de la commande n'est pas`CREATE_COMPLETE`.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name --query Stacks[].StackStatus --output text
   ```

   iii. Récupérez IDs les sous-réseaux publics qui ont été créés.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name \
       --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
    text
   ```

   L'exemple qui suit illustre un résultat.

   ```
   subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
   ```

iv. Activez l'option d'attribution automatique d'adresses `IPv6` pour les sous-réseaux publics qui ont été créés.

```
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

v. Récupérez les noms des sous-réseaux et des groupes de sécurité créés par le modèle à partir de la AWS CloudFormation pile déployée et stockez-les dans des variables pour une utilisation ultérieure.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --output
 text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --stack-
name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --output
 text)

subnets=${public_subnets},${private_subnets}
```

c. Créez un rôle IAM de cluster et attachez-y la politique gérée par Amazon EKS IAM requise. Les clusters Kubernetes gérés par Amazon EKS appellent d'autres AWS services en votre nom afin de gérer les ressources que vous utilisez avec le service.

i. Exécutez la commande suivante pour créer un fichier `eks-cluster-role-trust-policy.json` :

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

ii. Exécutez la commande suivante pour définir une variable pour votre nom de rôle. Vous pouvez remplacer *myAmazonEKSClusterRole* par n'importe quel nom que vous choisissez.

```
export cluster_role_name=myAmazonEKSClusterRole
```

iii. Créez le rôle.

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

iv. Récupérez l'ARN du rôle IAM et le stockez dans une variable pour une étape ultérieure.

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

v. Attachez la politique IAM gérée par Amazon EKS au rôle.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

d. Créer votre cluster.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $CLUSTER_IAM_ROLE --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

i. REMARQUE : il se peut que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins

deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

La création du cluster prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante tant que le résultat de la commande ne l'est pas`ACTIVE`.

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

e. Créez ou mettez à jour un fichier `kubeconfig` pour votre cluster afin de pouvoir communiquer avec votre cluster.

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

Par défaut, le `config` fichier est créé dans `~/.kube` ou la configuration du nouveau cluster est ajoutée à un `config` fichier existant dans`~/.kube`.

f. Créez un rôle IAM de nœud.

i. Exécutez la commande suivante pour créer un fichier `vpc-cni-ipv6-policy.json` :

```
cat >vpc-cni-ipv6-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:network-interface/*"
```

```
            ]
        }
    ]
}
EOF
```

ii. Créez la politique IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

iii. Exécutez la commande suivante pour créer un fichier `node-role-trust-relationship.json` :

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

iv. Exécutez la commande suivante pour définir une variable pour votre nom de rôle. Vous pouvez remplacer *`AmazonEKSNodeRole`* par n'importe quel nom que vous choisissez.

```
export node_role_name=AmazonEKSNodeRole
```

v. Créez le rôle IAM.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

vi. Attachez la politique IAM au rôle IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Pour simplifier ce tutoriel, la politique est attachée à ce rôle IAM. Toutefois, dans un cluster de production, nous recommandons de rattacher la politique à un rôle IAM distinct. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

vii. Attachez deux politiques gérées IAM requises au rôle IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

viii. Récupérez l'ARN du rôle IAM et le stockez dans une variable pour une étape ultérieure.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. Créez un groupe de nœuds gérés.

i. Affichez les IDs sous-réseaux que vous avez créés à l'étape précédente.

```
echo $subnets
```

L'exemple qui suit illustre un résultat.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-0377963d69EXAMPLE,subnet-0c05f819
```

ii. Créez le groupe de nœuds. Remplacez *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE*, et *0c05f819d5EXAMPLE* par les valeurs renvoyées dans la sortie de l'étape précédente. Assurez-vous de supprimer les virgules entre les sous-réseaux IDs de la

sortie précédente dans la commande suivante. Vous pouvez remplacer *t3.medium* par tout type d'instance du système AWS Nitro.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

La création du groupe de nœuds prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante tant que le résultat renvoyé ne l'est pas`ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

h. Vérifiez que les `IPv6` adresses sont attribuées aux pods par défaut dans la `IP` colonne.

```
kubectl get pods -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                     NODE                                            NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
```

```
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

i. Vérifiez que les adresses `IPv6` sont attribuées aux services par défaut dans la colonne `IP`.

```
kubectl get services -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

j. (Facultatif) Déployez un exemple d'application ou déployez le AWS Load Balancer Controller et un exemple d'application pour équilibrer la charge des applications HTTP the section called "Répartition de la charge des applications" ou le trafic réseau avec the section called "Répartition de charge réseau" les `IPv6` pods.

k. Une fois que vous avez terminé d'utiliser le cluster et les nœuds que vous avez créés pour ce didacticiel, vous devez nettoyer les ressources que vous avez créées à l'aide des commandes suivantes. Assurez-vous de n'utiliser aucune ressource en dehors de ce didacticiel avant de les supprimer.

   i. Si vous effectuez cette étape dans un interpréteur de commandes différent de celui dans lequel vous avez effectué les étapes précédentes, définissez les valeurs de toutes les variables utilisées dans les étapes précédentes, en les *`example values`* remplaçant par les valeurs que vous avez spécifiées lorsque vous avez effectué les étapes précédentes. Si vous effectuez cette étape dans le même shell que celui dans lequel vous avez effectué les étapes précédentes, passez à l'étape suivante.

```
export region_code=region-code
export vpc_stack_name=my-eks-ipv6-vpc
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
export node_role_name=AmazonEKSNodeRole
export cluster_role_name=myAmazonEKSClusterRole
```

   ii. Supprimez votre groupe de nœuds.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name
```

La suppression prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante si une sortie est renvoyée.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query
 nodegroups --output text
```

iii. Supprimez le cluster.

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

La suppression du cluster prend quelques minutes. Avant de continuer, assurez-vous que le cluster est supprimé avec la commande suivante.

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

Ne passez pas à l'étape suivante tant que votre sortie n'est pas similaire à la sortie suivante.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

iv. Supprimez les ressources IAM que vous avez créées. Remplacez *AmazonEKS_CNI_IPv6_Policy* par le nom que vous avez choisi, si vous avez choisi un nom différent de celui utilisé dans les étapes précédentes.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws: iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

v. Supprimez la AWS CloudFormation pile qui a créé le VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

Activer l'accès Internet sortant pour les Pods

S'applique à : nœuds Linux `IPv4` Fargate, nœuds Linux avec instances Amazon EC2

Si vous avez déployé votre cluster à l'aide de la `IPv6` famille, les informations de cette rubrique ne s'appliquent pas à votre cluster, car les `IPv6` adresses ne sont pas traduites par le réseau. Pour plus d'informations sur l'utilisation des adresses `IPv6` avec votre cluster, veuillez consulter the section called "IPv6".

Par défaut, chaque pod de votre cluster se voit attribuer une `IPv4` adresse privée provenant d'un bloc de routage inter-domaines sans classe (CIDR) associé au VPC dans lequel le pod est déployé. Les pods d'un même VPC communiquent entre eux en utilisant ces adresses IP privées comme points de terminaison. Lorsqu'un pod communique avec une `IPv4` adresse ne figurant pas dans un bloc CIDR associé à votre VPC, le plugin Amazon VPC CNI (pour Linux ou Windows) traduit l'adresse `IPv4` du pod en adresse `IPv4` privée principale de l'interface elastic network principale du nœud sur lequel le pod s'exécute, par défaut*.

> **Note**
>
> Pour les nœuds Windows, des détails supplémentaires doivent être pris en compte. Par défaut, le plug-in VPC CNI pour Windows est défini avec une configuration réseau dans laquelle le trafic vers une destination au sein du même VPC est exclu pour le SNAT. Cela signifie que le SNAT est désactivé pour la communication interne du VPC et que l'adresse IP allouée à un Pod est routable à l'intérieur du VPC. Mais le trafic vers une destination en dehors du VPC possède l'adresse IP du pod source transmise à l'adresse IP principale de l'instance ENI. Cette configuration par défaut pour Windows garantit que le pod peut accéder aux réseaux extérieurs à votre VPC de la même manière que l'instance hôte.

Compte tenu de ce comportement :

- Vos pods ne peuvent communiquer avec les ressources Internet que si le nœud sur lequel ils s'exécutent possède une adresse IP publique ou élastique et se trouve dans un sous-réseau public. La table de routage associée à un sous-réseau public possède une route vers une

passerelle Internet. Nous recommandons de déployer les nœuds dans des sous-réseaux privés, dans la mesure du possible.

- Pour les versions du plug-in antérieures`1.8.0`, les ressources qui se trouvent dans des réseaux ou VPCs qui sont connectées à votre VPC de cluster via le peering VPC, un VPC de transit ou Direct AWS Connect ne peuvent pas établir de communication avec vos Pods via des interfaces réseau élastiques secondaires. Vos Pods peuvent toutefois établir une communication avec ces ressources et recevoir des réponses de leur part.

Si l'une des affirmations suivantes est vraie dans votre environnement, modifiez la configuration par défaut à l'aide de la commande suivante.

- Vous disposez de ressources situées dans des réseaux ou VPCs connectées à votre VPC de cluster via le peering VPC, un VPC de transit ou Direct AWS Connect qui doivent établir une communication avec vos pods à l'aide d'une `IPv4` adresse et la version de votre plugin est antérieure à. `1.8.0`
- Vos pods se trouvent dans un sous-réseau privé et doivent communiquer vers Internet. Le sous-réseau dispose d'une route vers une passerelle NAT.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> Les variables de configuration `AWS_VPC_K8S_CNI_EXTERNALSNAT` et `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` CNI ne s'appliquent pas aux nœuds Windows. La désactivation du SNAT n'est pas prise en charge sous Windows. Quant à l'exclusion d'une liste `IPv4` CIDRs de du SNAT, vous pouvez la définir en spécifiant le `ExcludedSnatCIDRs` paramètre dans le script de démarrage Windows. Pour plus d'informations sur ce paramètre, veuillez consulter la rubrique the section called "Paramètres de configuration du script d'amorçage".

Réseau hôte

* Si la spécification d'un pod contient `hostNetwork=true` (la valeur par défaut est`false`), son adresse IP n'est pas traduite vers une adresse différente. C'est le cas du plugin Amazon VPC CNI pour Kubernetes Pods qui s'exécutent par défaut sur votre cluster. `kube-proxy` Pour ces pods,

l'adresse IP est identique à l'adresse IP principale du nœud, de sorte que l'adresse IP du pod n'est pas traduite. Pour plus d'informations sur les `hostNetwork` paramètres d'un pod, consultez la section PodSpec v1 core dans la référence de l'API Kubernetes.

Limitez le trafic des pods grâce aux politiques réseau Kubernetes

Par défaut, Kubernetes n'impose aucune restriction concernant les adresses IP, les ports ou les connexions entre les pods de votre cluster ou entre vos pods et les ressources d'un autre réseau. Vous pouvez utiliser la politique réseau de Kubernetes pour restreindre le trafic réseau à destination et en provenance de vos pods. Pour plus d'informations, consultez la section Politiques réseau dans la documentation de Kubernetes.

Si vous disposez d'une version `1.13` ou d'une version antérieure du plug-in Amazon VPC CNI pour Kubernetes sur votre cluster, vous devez implémenter une solution tierce pour appliquer les politiques réseau Kubernetes à votre cluster. La version `1.14` ou ultérieure du plugin peut implémenter des politiques réseau, vous n'avez donc pas besoin d'utiliser une solution tierce. Dans cette rubrique, vous apprendrez à configurer votre cluster pour utiliser la politique réseau Kubernetes sur votre cluster sans utiliser de module complémentaire tiers.

Les politiques réseau du plug-in Amazon VPC CNI pour Kubernetes sont prises en charge dans les configurations suivantes.

- Version 1.14 ou ultérieure du plugin Amazon VPC CNI pour Kubernetes sur votre cluster.
- Cluster configuré pour les adresses `IPv4` ou `IPv6`.
- Vous pouvez utiliser des politiques réseau avec des groupes de sécurité pour les pods. Grâce aux stratégies réseau, vous pouvez contrôler toutes les communications au sein du cluster. Avec les groupes de sécurité pour Pods, vous pouvez contrôler l'accès aux AWS services depuis les applications d'un Pod.
- Vous pouvez utiliser les stratégies réseau avec mise en réseau personnalisée et délégation de préfixes.

Considérations

Architecture

- Lorsque vous appliquez les politiques réseau du plug-in Amazon VPC CNI pour Kubernetes à votre cluster à l'aide du plug-in Amazon VPC CNI pour Kubernetes, vous pouvez appliquer les politiques aux nœuds Amazon Linux uniquement. EC2 Vous ne pouvez pas appliquer les politiques aux nœuds Fargate ou Windows.

- Les politiques réseau s'appliquent uniquement à l'une `IPv4` ou l'autre `IPv6` des adresses, mais pas aux deux. Dans un `IPv4` cluster, le VPC CNI attribue une `IPv4` adresse aux pods et applique des politiques. `IPv4` Dans un `IPv6` cluster, le VPC CNI attribue une `IPv6` adresse aux pods et applique des politiques. `IPv6` Toutes les règles de politique `IPv4` réseau appliquées à un `IPv6` cluster sont ignorées. Toutes les règles de politique `IPv6` réseau appliquées à un `IPv4` cluster sont ignorées.

Politiques du réseau

- Les politiques réseau ne sont appliquées qu'aux pods faisant partie d'un déploiement. Les politiques réseau ne peuvent pas être appliquées aux pods autonomes qui ne disposent pas d'un `metadata.ownerReferences` ensemble.
- Vous pouvez appliquer plusieurs politiques réseau au même Pod. Lorsque deux politiques ou plus qui sélectionnent le même Pod sont configurées, toutes les politiques sont appliquées au Pod.
- Le nombre maximum de combinaisons uniques de ports pour chaque protocole dans chaque `egress:` sélecteur `ingress:` d'une politique réseau est de 24.
- Quel que soit votre service Kubernetes, le port de service doit être le même que le port de conteneur. Si vous utilisez des ports nommés, utilisez également le même nom dans les spécifications du service.

Migration

- Si votre cluster utilise actuellement une solution tierce pour gérer les politiques réseau Kubernetes, vous pouvez utiliser ces mêmes politiques avec le plug-in Amazon VPC CNI pour Kubernetes. Vous devez toutefois supprimer votre solution existante afin qu'elle ne gère pas les mêmes politiques.

Montage

- La fonctionnalité de stratégie réseau crée et nécessite une définition de ressource personnalisée (CRD) de `PolicyEndpoint` appelée `policyendpoints.networking.k8s.aws`. Les objets du `PolicyEndpoint` de la ressource personnalisée sont gérés par Amazon EKS. Vous ne devez ni modifier ni supprimer ces ressources.
- Si vous exécutez des pods qui utilisent les informations d'identification IAM du rôle d'instance ou si vous vous connectez à l' EC2 IMDS, veillez à vérifier les politiques réseau susceptibles de bloquer l'accès à l' EC2 IMDS. Vous devrez peut-être ajouter une politique réseau pour autoriser l'accès à

l' EC2 IMDS. Pour plus d'informations, consultez la section Métadonnées de l'instance et données utilisateur dans le guide de EC2 l'utilisateur Amazon.

Les pods qui utilisent des rôles IAM pour les comptes de service ou EKS Pod Identity n'accèdent pas à EC2 IMDS.

- Le plug-in Amazon VPC CNI pour Kubernetes n'applique pas de politiques réseau aux interfaces réseau supplémentaires pour chaque pod, mais uniquement à l'interface principale pour chaque pod (). `eth0` Cela concerne les architectures suivantes :
  - `IPv6` pods dont la variable `ENABLE_V4_EGRESS` est définie sur `true`. Cette variable permet à la fonction de `IPv4` sortie de connecter les IPv6 pods à des `IPv4` points de terminaison tels que ceux situés en dehors du cluster. La fonction de `IPv4` sortie fonctionne en créant une interface réseau supplémentaire avec une adresse de boucle locale. IPv4
  - Lorsque vous utilisez des plugins réseau chaînés tels que Multus. Comme ces plug-ins ajoutent des interfaces réseau à chaque module, les politiques réseau ne sont pas appliquées aux plug-ins réseau enchaînés.

Limitez le trafic réseau des Pod grâce aux politiques réseau de Kubernetes

Vous pouvez utiliser une politique réseau Kubernetes pour restreindre le trafic réseau à destination et en provenance de vos pods. Pour plus d'informations, consultez la section Politiques réseau dans la documentation de Kubernetes.

Pour utiliser cette fonctionnalité, vous devez configurer les éléments suivants :

1. Configurez l'application des politiques au démarrage du Pod. Vous le faites dans le `aws-node` conteneur du VPC CNI. `DaemonSet`
2. Activez le paramètre de politique réseau pour le module complémentaire.
3. Configurez votre cluster pour utiliser la politique réseau Kubernetes

Avant de commencer, passez en revue les considérations. Pour de plus amples informations, veuillez consulter the section called “Considérations”.

Prérequis

Les conditions requises pour cette fonctionnalité sont les suivantes :

Version minimale du cluster

Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route. Le cluster doit exécuter l'une des versions de Kubernetes et de la plateforme répertoriées dans le tableau suivant. Notez que toutes les versions de Kubernetes et de plate-forme ultérieures à celles répertoriées sont également prises en charge. Vous pouvez vérifier votre version actuelle de Kubernetes *my-cluster* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Version de Kubernetes	Version de la plateforme
1.27.4	eks.5
1.26.7	eks.6

Version minimale du VPC CNI

Version 1.14 ou version ultérieure du plugin Amazon VPC CNI pour Kubernetes sur votre cluster. Vous pouvez voir la version que vous utilisez actuellement à l'aide de la commande suivante.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Si votre version est antérieure à 1.14, consultez la section the section called "Mise à jour (module complémentaire EKS)" pour mettre à jour vers la version 1.14 ou ultérieure.

Version minimale du noyau Linux

Vos nœuds doivent avoir la version du noyau Linux 5.10 ou ultérieure. Vous pouvez vérifier votre version du noyau avec uname -r. Si vous utilisez les dernières versions d'Amazon Linux optimisé pour Amazon EKS, d'Amazon Linux accéléré optimisé pour Amazon EKS et de Bottlerocket AMIs, ils disposent déjà de la version de noyau requise. AMIs

La version v20231116 ou ultérieure de l'AMI Amazon Linux accélérée et optimisée pour Amazon EKS dispose de la version 5.10 du noyau.

Étape 1 : configurer l'application des politiques au démarrage du Pod

Le plug-in Amazon VPC CNI pour Kubernetes configure les politiques réseau pour les pods en parallèle avec le provisionnement des pods. Jusqu'à ce que toutes les politiques soient configurées pour le nouveau pod, les conteneurs du nouveau pod commenceront par une politique d'autorisation par défaut. C'est ce qu'on appelle le mode standard. Une politique d'autorisation par défaut signifie que tout le trafic entrant et sortant est autorisé à destination et en provenance des nouveaux modules. Par exemple, aucune règle de pare-feu ne sera appliquée aux pods (tout le trafic est autorisé) tant que le nouveau pod ne sera pas mis à jour avec les politiques actives.

Lorsque la `NETWORK_POLICY_ENFORCING_MODE` variable est définie sur`strict`, les pods qui utilisent le VPC CNI commencent par une politique de refus par défaut, puis les politiques sont configurées. C'est ce qu'on appelle le mode strict. En mode strict, vous devez disposer d'une politique réseau pour chaque point de terminaison auquel vos pods doivent accéder dans votre cluster. Notez que cette exigence s'applique aux pods CoreDNS. La politique de refus par défaut n'est pas configurée pour les pods dotés d'un réseau hôte.

Vous pouvez modifier la politique réseau par défaut en définissant la variable `NETWORK_POLICY_ENFORCING_MODE` d'environnement sur `strict` dans le `aws-node` conteneur du VPC CNI. `DaemonSet`

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

Étape 2 : activer le paramètre de politique réseau pour le module complémentaire

La fonctionnalité de politique réseau utilise le port `8162` du nœud pour les métriques par défaut. En outre, la fonctionnalité utilisait le port `8163` pour les sondes d'intégrité. Si vous exécutez une autre application sur les nœuds ou dans les pods qui doivent utiliser ces ports, l'application ne s'exécutera pas. Dans la version VPC CNI `v1.14.1` ou ultérieure, vous pouvez modifier ces ports.

Utilisez la procédure suivante pour activer le paramètre de politique réseau pour le module complémentaire.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.

3. Choisissez l'onglet Modules complémentaires.
4. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).
5. Sur la `Amazon VPC CNI` page de configuration :
   a. Sélectionnez une version `v1.14.0-eksbuild.3` ou une version ultérieure dans la liste des versions.
   b. Sélectionnez Paramètres de configuration facultatifs.
   c. Saisissez la clé JSON `"enableNetworkPolicy":` et valeur `"true"` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`.

      Dans l'exemple suivant, la fonctionnalité de politique réseau est activée et les métriques et les sondes de santé sont définies sur les numéros de port par défaut :

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via`helm`, vous pouvez mettre à jour la configuration pour modifier les ports.

1. Exécutez la commande suivante pour modifier les ports. Définissez le numéro de port dans la valeur de la clé `nodeAgent.metricsBindAddr` ou de la clé `nodeAgent.healthProbeBindAddr`.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. Ouvrez le `aws-node DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Remplacez les numéros de port dans les arguments de commande suivants pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste daemonset `aws-node` du VPC CNI.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

Étape 3 : monter le système de fichiers Berkeley Packet Filter (BPF) sur vos nœuds

Vous devez monter le système de fichiers Berkeley Packet Filter (BPF) sur chacun de vos nœuds.

> **Note**
>
> Si votre cluster est une version `1.27` ou une version ultérieure, vous pouvez ignorer cette étape car tous les appareils Amazon Linux et Bottlerocket optimisés AMIs pour Amazon EKS `1.27` ou ultérieurs disposent déjà de cette fonctionnalité.
> Pour toutes les autres versions de cluster, si vous mettez à niveau Amazon Linux optimisé pour Amazon EKS vers la version `v20230703` ou ultérieure ou vous mettez à niveau l'AMI Bottlerocket vers la version `v1.0.2` ou ultérieure, vous pouvez ignorer cette étape.

1. Montez le système de fichiers Berkeley Packet Filter (BPF) sur chacun de vos nœuds.

```
sudo mount -t bpf bpffs /sys/fs/bpf
```

2. Ajoutez ensuite la même commande à vos données utilisateur dans votre modèle de lancement pour vos groupes Amazon EC2 Auto Scaling.

Étape 4 : configurer votre cluster pour utiliser les politiques réseau Kubernetes

Vous pouvez le définir pour un module complémentaire Amazon EKS ou un module complémentaire autogéré.

Module complémentaire d'Amazon EKS

À l'aide de la AWS CLI, vous pouvez configurer le cluster pour utiliser les politiques réseau Kubernetes en exécutant la commande suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":
 "true"}'
```

Pour le configurer à l'aide AWS de la console de gestion, suivez les étapes ci-dessous :

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.
3. Choisissez l'onglet Modules complémentaires.
4. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).
5. Sur la `Amazon VPC CNI` page de configuration :
   a. Sélectionnez une version `v1.14.0-eksbuild.3` ou une version ultérieure dans la liste des versions.
   b. Sélectionnez Paramètres de configuration facultatifs.
   c. Saisissez la clé JSON `"enableNetworkPolicy":` et valeur `"true"` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`. L'exemple suivant montre que la politique réseau est activée :

      ```
      { "enableNetworkPolicy": "true" }
      ```

      La capture d'écran suivante montre un exemple de ce scénario.

EKS > Clusters > > Add-on > vpc-cni > Edit add-on

Configure Amazon VPC CNI

Amazon VPC CNI Info

Listed by	Category	Status
	networking	Active

Version

Select the version for this add-on.

v1.17.1-eksbuild.1

Select IAM role

Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.

Optional configuration settings

Add-on configuration schema

Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$ref": "#/definitions/VpcCni",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "definitions": {
    "Affinity": {
      "type": [
        "object",
        "null"
      ]
    },
    "EniConfig": {
      "additionalProperties": false,
```

Configuration values Info

Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 { "enableNetworkPolicy": "true" }
```

Module complémentaire autogéré

Helm

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via`helm`, vous pouvez mettre à jour la configuration pour activer la politique réseau.

1. Exécutez la commande suivante pour activer la stratégie réseau.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Ouvrez le `amazon-vpc-cni` `ConfigMap` dans votre éditeur.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. Ajoutez la ligne suivante dans les `data` de `ConfigMap`.

```
enable-network-policy-controller: "true"
```

   Une fois que vous avez ajouté la ligne, vous `ConfigMap` devriez ressembler à l'exemple suivant.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

3. Ouvrez le `aws-node` `DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

   a. Remplacez `false` par `true` dans l'argument de commande `--enable-network-policy=false` pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste daemonset du `aws-node` VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

Étape 5. Étapes suivantes

Une fois la configuration terminée, vérifiez que les `aws-node` pods sont en cours d'exécution sur votre cluster.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

L'exemple qui suit illustre un résultat.

```
aws-node-gmqp7                                          2/2     Running   1 (24h ago)
 24h
aws-node-prnsh                                          2/2     Running   1 (24h ago)
 24h
```

Il y a 2 conteneurs dans les `aws-node` capsules dans les versions `1.14` et ultérieures. Dans les versions précédentes et si la stratégie réseau est désactivée, il n'y a qu'un seul conteneur dans les pods du `aws-node`.

Vous pouvez désormais déployer des politiques réseau Kubernetes sur votre cluster.

Pour implémenter les politiques réseau Kubernetes, vous devez créer des `NetworkPolicy` objets Kubernetes et les déployer sur votre cluster. `NetworkPolicy`les objets sont limités à un espace de noms. Vous implémentez des politiques pour autoriser ou refuser le trafic entre les pods en fonction des sélecteurs d'étiquettes, des espaces de noms et des plages d'adresses IP. Pour plus d'informations sur la création d'`NetworkPolicy`objets, consultez la section Politiques réseau dans la documentation de Kubernetes.

L'application des `NetworkPolicy` objets Kubernetes est mise en œuvre à l'aide du filtre de paquets étendu de Berkeley (eBPF). Concernant les implémentations basées sur les `iptables`, il offre des caractéristiques de latence et de performance plus faibles, notamment une utilisation réduite du processeur et l'évitement des recherches séquentielles. En outre, les sondes eBPF permettent d'accéder à des données contextuelles qui aident à résoudre les problèmes complexes au niveau du noyau et à améliorer l'observabilité. Amazon EKS prend en charge un exportateur basé sur EBPF qui utilise les sondes pour enregistrer les résultats des politiques sur chaque nœud et exporter

les données vers des collecteurs de journaux externes afin de faciliter le débogage. Pour plus d'informations, consultez la documentation eBPF.

Désactiver les politiques réseau Kubernetes pour le trafic réseau Amazon EKS Pod

Désactivez les politiques réseau Kubernetes pour arrêter de restreindre le trafic réseau Amazon EKS Pod

1. Répertoriez toutes les politiques réseau Kubernetes.

```
kubectl get netpol -A
```

2. Supprimez chaque politique réseau Kubernetes. Vous devez supprimer toutes les politiques réseau avant de les désactiver.

```
kubectl delete netpol <policy-name>
```

3. Ouvrez le aws-node DaemonSet dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

4. Remplacez `true` par `false` dans l'argument de commande `--enable-network-policy=true` pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste daemonset du `aws-node` VPC CNI.

```
      - args:
         - --enable-network-policy=true
```

Résolution des problèmes liés aux politiques réseau Kubernetes pour Amazon EKS

Il s'agit du guide de dépannage relatif à la fonctionnalité de politique réseau de l'Amazon VPC CNI.

Ce guide couvre les sujets suivants :

- Informations d'installation, autorisations CRD et RBAC the section called "Nouveau `policyendpoints` CRD et nouvelles autorisations"
- Journaux à examiner lors du diagnostic des problèmes de politique réseau the section called "Journaux de stratégie réseau"
- Exécution de la collection d'outils du SDK eBPF pour résoudre les problèmes

- Problèmes connus et solutions the section called "Problèmes connus et solutions"

> **Note**
>
> Notez que les politiques réseau ne sont appliquées qu'aux pods créés par Kubernetes Deployments. Pour en savoir plus sur les limites des politiques réseau du VPC CNI, consultez. the section called "Considérations"

Vous pouvez résoudre les problèmes et étudier les connexions réseau qui utilisent des politiques réseau en lisant les journaux des politiques réseau et en exécutant les outils du SDK eBPF.

Nouveau **`policyendpoints`** CRD et nouvelles autorisations

- CARTE : `policyendpoints.networking.k8s.aws`
- API Kubernetes : appelée `apiservice v1.networking.k8s.io`
- Ressource Kubernetes : `Kind: NetworkPolicy`
- RBAC : `ClusterRole` appelé `aws-node` (VPC CNI)`ClusterRole`, `eks:network-policy-controller` appelé (contrôleur de politique réseau dans le plan de contrôle du cluster EKS)

Pour la politique réseau, le VPC CNI crée un nouveau `CustomResourceDefinition` (CRD) appelé. `policyendpoints.networking.k8s.aws` Le CNI du VPC doit être autorisé à créer le CRD et à créer CustomResources (CR) celui-ci et l'autre CRD installé par le VPC CNI (). `eniconfigs.crd.k8s.amazonaws.com` Les deux CRDs sont disponibles dans le `crds.yaml`fichier sur GitHub. Plus précisément, le CNI du VPC doit disposer des autorisations verbales « get », « list » et « watch » pour. `policyendpoints`

La politique réseau Kubernetes fait partie de l'`apiservice`appel`v1.networking.k8s.io`, et elle se trouve `apiversion: networking.k8s.io/v1` dans vos fichiers YAML de politique. Le VPC CNI `DaemonSet` doit être autorisé à utiliser cette partie de l'API Kubernetes.

Les autorisations VPC CNI se trouvent dans un appel. `ClusterRole aws-node` Notez que les `ClusterRole` objets ne sont pas regroupés dans des espaces de noms. Ce qui suit montre le `aws-node` schéma d'un cluster :

```
kubectl get clusterrole aws-node -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/instance: aws-vpc-cni
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: aws-node
    app.kubernetes.io/version: v1.19.4
    helm.sh/chart: aws-vpc-cni-1.19.4
    k8s-app: aws-node
  name: aws-node
rules:
- apiGroups:
  - crd.k8s.amazonaws.com
  resources:
  - eniconfigs
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - list
  - watch
```

```
  - get
- apiGroups:
  - ""
  - events.k8s.io
  resources:
  - events
  verbs:
  - create
  - patch
  - list
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
- apiGroups:
  - vpcresources.k8s.aws
  resources:
  - cninodes
  verbs:
  - get
  - list
  - watch
  - patch
```

En outre, un nouveau contrôleur s'exécute dans le plan de contrôle de chaque cluster EKS. Le contrôleur utilise les autorisations de l'`ClusterRoleappeléeks:network-policy-controller`. Ce qui suit montre le `eks:network-policy-controller` schéma d'un cluster :

```
kubectl get clusterrole eks:network-policy-controller -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  labels:
    app.kubernetes.io/name: amazon-network-policy-controller-k8s
  name: eks:network-policy-controller
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
```

```
  - policyendpoints/finalizers
  verbs:
  - update
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
  - patch
  - update
- apiGroups:
  - networking.k8s.io
  resources:
  - networkpolicies
  verbs:
  - get
  - list
  - patch
  - update
  - watch
```

Journaux de stratégie réseau

Chaque décision prise par le VPC CNI quant à savoir si les connexions sont autorisées ou refusées par une politique réseau est enregistrée dans des journaux de flux. Les journaux de stratégie réseau de chaque nœud incluent les journaux de flux pour chaque pod doté d'une stratégie réseau. Les journaux de stratégie réseau sont stockés sur `/var/log/aws-routed-eni/network-policy-agent.log`. L'exemple suivant est extrait d'un fichier `network-policy-agent.log` :

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

Les journaux de politique réseau sont désactivés par défaut. Pour activer les journaux de politique réseau, procédez comme suit :

> **Note**
>
> Les journaux de politique réseau nécessitent 1 vCPU supplémentaire pour le `aws-network-policy-agent` conteneur dans le manifeste VPC CNI. `aws-node DaemonSet`

Module complémentaire d'Amazon EKS

AWS Management Console

a. Ouvrez la console Amazon EKS.

b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.

c. Choisissez l'onglet Modules complémentaires.

d. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).

e. Sur la *`Amazon VPC CNI`* page de configuration :

   i. Sélectionnez une version `v1.14.0-eksbuild.3` ou ultérieure dans la liste déroulante Version.

   ii. Sélectionnez Paramètres de configuration facultatifs.

   iii. Saisissez la clé JSON de niveau supérieur de l'`'"nodeAgent":` et la valeur est un objet avec une clé `"enablePolicyEventLogs":` et une valeur de `"true"` dansValeurs de configuration. Le texte obtenu doit être un objet JSON valide. L'exemple suivant montre la politique réseau et les journaux de stratégie réseau sont activés, et les journaux de stratégie réseau sont envoyés à CloudWatch Logs :

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

La capture d'écran suivante montre un exemple de ce scénario.

EKS > Clusters > > Add-on > vpc-cni > Edit add-on

Configure Amazon VPC CNI

Amazon VPC CNI Info

Listed by

Category
networking

Status
Active

Version
Select the version for this add-on.

v1.17.1-eksbuild.1

Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.

▼ Optional configuration settings

Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$ref": "#/definitions/VpcCni",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "definitions": {
    "Affinity": {
      "type": [
        "object",
        "null"
      ]
    },
    "EniConfig": {
      "additionalProperties": false,
```

Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 {
2     "enableNetworkPolicy": "true",
3     "nodeAgent": {
4         "enablePolicyEventLogs": "true"
5     }
6 }
```

AWS CLI

a. Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

Module complémentaire autogéré

Helm

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via`helm`, vous pouvez mettre à jour la configuration pour écrire les journaux de politique réseau.

a. Exécutez la commande suivante pour activer la stratégie réseau.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via`kubectl`, vous pouvez mettre à jour la configuration pour écrire les journaux de politique réseau.

a. Ouvrez le `aws-node DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Remplacez le `false` par `true` dans l'argument de commande `--enable-policy-event-logs=false` `args:` dans le `aws-network-policy-agent` conteneur du manifeste VPC `aws-node DaemonSet` CNI.

```
     - args:
        - --enable-policy-event-logs=true
```

Envoyer des journaux de politique réseau à Amazon CloudWatch Logs

Vous pouvez surveiller les journaux de politique réseau à l'aide de services tels qu'Amazon CloudWatch Logs. Vous pouvez utiliser les méthodes suivantes pour envoyer les journaux de politique réseau à CloudWatch Logs.

Pour les clusters EKS, les journaux des politiques seront situés en dessous `/aws/eks/cluster-name/cluster/` et pour les clusters K8S autogérés, les journaux seront placés en dessous. `/aws/k8s-cluster/cluster/`

Envoyez des journaux de politique réseau avec le plugin Amazon VPC CNI pour Kubernetes

Si vous activez la stratégie réseau, un deuxième conteneur est ajouté aux pods du `aws-node` pour unagent de nœud. Cet agent de nœud peut envoyer les journaux de politique réseau à CloudWatch Logs.

> **Note**
>
> Seuls les journaux de stratégie réseau sont envoyés par l'agent de nœud. Les autres journaux créés par le VPC CNI ne sont pas inclus.

Prérequis

- Ajoutez les autorisations suivantes sous forme de strophe ou de stratégie distincte au rôle IAM que vous utilisez pour le VPC CNI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Module complémentaire d'Amazon EKS

AWS Management Console

a. Ouvrez la console Amazon EKS.
b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.
c. Choisissez l'onglet Modules complémentaires.
d. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).
e. Sur la *Amazon VPC CNI* page de configuration :
   i. Sélectionnez une version `v1.14.0-eksbuild.3` ou ultérieure dans la liste déroulante Version.
   ii. Sélectionnez Paramètres de configuration facultatifs.
   iii. Saisissez la clé JSON de niveau supérieur de l'`"nodeAgent":` et la valeur est un objet avec une clé `"enableCloudWatchLogs":` et une valeur de `"true"` dansValeurs de configuration. Le texte obtenu doit être un objet JSON valide. L'exemple suivant montre la politique réseau, les journaux de stratégie réseau sont activés et les journaux sont envoyés à CloudWatch Logs :

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

La capture d'écran suivante montre un exemple de ce scénario.

EKS > Clusters > > Add-on > vpc-cni > Edit add-on

Configure Amazon VPC CNI

Amazon VPC CNI Info

Listed by

Category
networking

Status
Active

Version
Select the version for this add-on.

v1.17.1-eksbuild.1

Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.

▼ Optional configuration settings

Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.

```
{
  "$ref": "#/definitions/VpcCni",
  "$schema": "http://json-schema.org/draft-06/schema#",
  "definitions": {
    "Affinity": {
      "type": [
        "object",
        "null"
      ]
    },
    "EniConfig": {
      "additionalProperties": false,
```

Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.

```
1 {
2     "enableNetworkPolicy": "true",
3     "nodeAgent": {
4         "enablePolicyEventLogs": "true",
5         "enableCloudWatchLogs": "true"
6     }
7 }
```

AWS CLI

a. Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

Module complémentaire autogéré

Casque

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via`helm`, vous pouvez mettre à jour la configuration pour envoyer les journaux de politique réseau à Logs. CloudWatch

a. Exécutez la commande suivante pour activer les journaux de politique réseau et les envoyer à CloudWatch Logs.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-
vpc-cni
```

kubectl

a. Ouvrez le `aws-node` `DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Remplacez le `false` par `true` dans deux arguments de commande `--enable-policy-event-logs=false` et `--enable-cloudwatch-logs=false` `args:` dans le `aws-network-policy-agent` conteneur du manifeste VPC `aws-node` `DaemonSet` CNI.

```
     - args:
        - --enable-policy-event-logs=true
        - --enable-cloudwatch-logs=true
```

Envoyer des journaux de politique réseau avec un Fluent Bit **`DaemonSet`**

Si vous utilisez Fluent Bit dans un `DaemonSet` pour envoyer des journaux depuis vos nœuds, vous pouvez ajouter une configuration pour inclure les journaux de politique réseau à partir des politiques réseau. Vous pouvez utiliser l'exemple de configuration suivant :

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

SDK eBPF inclus

Le plug-in Amazon VPC CNI pour Kubernetes installe une collection d'outils du SDK eBPF sur les nœuds. Vous pouvez utiliser les outils du SDK eBPF pour identifier les problèmes liés aux politiques réseau. Par exemple, la commande suivante répertorie les programmes qui s'exécutent sur le nœud.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Pour exécuter cette commande, vous pouvez utiliser n'importe quelle méthode pour vous connecter au nœud.

Problèmes connus et solutions

Les sections suivantes décrivent les problèmes connus liés à la fonctionnalité de politique réseau CNI d'Amazon VPC et à leurs solutions.

Journaux de politique réseau générés même si la valeur enable-policy-event-logs est définie sur false

Problème : EKS VPC CNI génère des journaux de politique réseau même lorsque le `enable-policy-event-logs` paramètre est défini sur. `false`

Solution : le `enable-policy-event-logs` paramètre désactive uniquement les journaux des « décisions » relatives aux politiques, mais il ne désactive pas tous les journaux de l'agent Network Policy. Ce comportement est documenté dans le aws-network-policy-agent fichier README sur

GitHub. Pour désactiver complètement la journalisation, vous devrez peut-être ajuster d'autres configurations de journalisation.

Problèmes liés au nettoyage de la carte des politiques réseau

Problème : les problèmes liés au réseau existent `policyendpoint` toujours et ne sont pas nettoyés après la suppression des pods.

Solution : Ce problème est dû à un problème lié à la version 1.19.3-eksbuild.1 du module complémentaire VPC CNI. Effectuez une mise à jour vers une version plus récente du module complémentaire VPC CNI pour résoudre ce problème.

Les politiques réseau ne sont pas appliquées

Problème : la fonctionnalité de politique réseau est activée dans le plug-in Amazon VPC CNI, mais les politiques réseau ne sont pas appliquées correctement.

Si vous définissez une politique réseau `kind: NetworkPolicy` et que cela n'affecte pas le pod, vérifiez que l'objet policyendpoint a été créé dans le même espace de noms que le pod. S'il n'y a aucun `policyendpoint` objet dans les espaces de noms, le contrôleur de politique réseau (qui fait partie du cluster EKS) n'a pas pu créer de règles de politique réseau à appliquer par l'agent de stratégie réseau (qui fait partie du CNI VPC).

Solution : La solution consiste à fixer les autorisations du VPC CNI (`ClusterRole:aws-node`) et du contrôleur de politique réseau (`ClusterRole:eks:network-policy-controller`) et à autoriser ces actions dans n'importe quel outil d'application des politiques tel que Kyverno. Assurez-vous que les politiques de Kyverno ne bloquent pas la création d'`policyendpoint`objets. Reportez-vous à la section précédente pour connaître les autorisations nécessaires dansthe section called "Nouveau `policyendpoints` CRD et nouvelles autorisations".

Les pods ne reviennent pas à l'état de refus par défaut après la suppression de la politique en mode strict

Problème : Lorsque les politiques réseau sont activées en mode strict, les pods démarrent avec une politique de refus par défaut. Une fois les politiques appliquées, le trafic est autorisé vers les points de terminaison spécifiés. Cependant, lorsque les politiques sont supprimées, le pod ne revient pas à l'état de refus par défaut mais passe à l'état d'autorisation par défaut.

Solution : Ce problème a été résolu dans la version 1.19.3 de VPC CNI, qui incluait la version 1.2.0 de l'agent de politique réseau. Après le correctif, lorsque le mode strict est activé, une fois les politiques supprimées, le pod reviendra à l'état de refus par défaut, comme prévu.

Groupes de sécurité pour la latence de démarrage des Pods

Problème : lorsque vous utilisez la fonctionnalité Security Groups for Pods dans EKS, la latence de démarrage des pods augmente.

Solution : La latence est due à la limitation du débit dans le contrôleur de ressources due à la limitation de l'`CreateNetworkInterface`API, que le contrôleur de ressources VPC utilise pour créer une branche ENIs pour les pods. Vérifiez les limites de l'API de votre compte pour cette opération et envisagez de demander une augmentation des limites si nécessaire.

FailedScheduling en raison d'une insuffisance de vpc.amazonaws.com/pod-eni

Problème : Impossible de planifier les pods avec le message d'erreur suivant : `FailedScheduling 2m53s (x28 over 137m) default-scheduler 0/5 nodes are available: 5 Insufficient vpc.amazonaws.com/pod-eni. preemption: 0/5 nodes are available: 5 No preemption victims found for incoming pod.`

Solution : Comme dans le cas du problème précédent, l'attribution de groupes de sécurité aux pods augmente la latence de planification des pods et elle peut dépasser le seuil CNI nécessaire pour ajouter chaque ENI, ce qui entraîne des échecs lors du démarrage des pods. Ce comportement est attendu lors de l'utilisation de groupes de sécurité pour les pods. Tenez compte des implications en matière de planification lors de la conception de votre architecture de charge de travail.

Problèmes de connectivité IPAM et défauts de segmentation

Problème : plusieurs erreurs se produisent, notamment des problèmes de connectivité IPAM, des demandes de régulation et des erreurs de segmentation :

- `Checking for IPAM connectivity …`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

Solution : Ce problème se produit si vous installez `systemd-udev` le fichier AL2 023, car le fichier est réécrit avec une politique d'interruption. Cela peut se produire lors de la mise à jour vers un autre `releasever` package doté d'un package mis à jour ou lors de la mise à jour manuelle du package lui-même. Évitez d'installer ou de mettre à jour `systemd-udev` sur les nœuds AL2 023.

Impossible de trouver le périphérique par son nom d'erreur

Problème : message d'erreur :
```
{"level":"error","ts":"2025-02-05T20:27:18.669Z","caller":"ebpf/
bpf_client.go:578","msg":"failed to find device by name eni9ea69618bf0: %!
w(netlink.LinkNotFoundError={0xc000115310})"}
```

Solution : ce problème a été identifié et résolu dans les dernières versions de l'agent de politique réseau Amazon VPC CNI (v1.2.0). Passez à la dernière version du VPC CNI pour résoudre ce problème.

Vulnérabilités CVE dans l'image Multus CNI

Problème : le rapport EKS ImageScan CVE amélioré identifie des vulnérabilités dans la version v4.1.4-eksbuild.2_thick de l'image Multus CNI.

Solution : passez à la nouvelle version de l'image Multus CNI et à la nouvelle image du Network Policy Controller, qui ne présentent aucune vulnérabilité. Le scanner peut être mis à jour pour corriger les vulnérabilités détectées dans la version précédente.

Flow Info : REFUSEZ les verdicts dans les logs

Problème : les journaux de politique réseau affichent les verdicts de refus :
```
{"level":"info","ts":"2024-11-25T13:34:24.808Z","logger":"ebpf-
client","caller":"events/events.go:193","msg":"Flow
Info: ","Src IP":"","Src Port":9096,"Dest IP":"","Dest
Port":56830,"Proto":"TCP","Verdict":"DENY"}
```

Solution : Ce problème a été résolu dans la nouvelle version du Network Policy Controller. Passez à la dernière version de la plateforme EKS pour résoudre les problèmes de journalisation.

Pod-to-pod problèmes de communication après la migration depuis Calico

Problème : Après la mise à niveau d'un cluster EKS vers la version 1.30 et le passage de Calico à Amazon VPC CNI pour la politique réseau, la pod-to-pod communication échoue lorsque les politiques réseau sont appliquées. La communication est rétablie lorsque les politiques réseau sont supprimées.

Solution : L'agent de politique réseau du VPC CNI ne peut pas avoir autant de ports spécifiés que Calico. Utilisez plutôt des plages de ports dans les politiques réseau. Le nombre maximum de

combinaisons uniques de ports pour chaque protocole dans chaque `egress:` sélecteur `ingress:` d'une politique réseau est de 24. Utilisez des plages de ports pour réduire le nombre de ports uniques et éviter cette limitation.

L'agent de stratégie réseau ne prend pas en charge les modules autonomes

Problème : les politiques réseau appliquées aux pods autonomes peuvent avoir un comportement incohérent.

Solution : L'agent Network Policy ne prend actuellement en charge que les pods déployés dans le cadre d'un déployement/d'un ensemble de réplicasets. Si des politiques réseau sont appliquées à des pods autonomes, le comportement peut présenter des incohérences. Ceci est documenté en haut de cette page, dans le the section called "Considérations" numéro et dans le aws-network-policy-agent GitHub numéro #327 sur GitHub. Déployez des pods dans le cadre d'un déploiement ou d'un réplicaset pour un comportement cohérent en matière de politique réseau.

Démonstration de la politique réseau d'Amazon EKS avec Stars

Cette démonstration permet de créer un service client, un service frontal et un service backend sur votre cluster Amazon EKS. La démonstration crée également une interface utilisateur graphique de gestion qui affiche les chemins d'entrée et de sortie disponibles entre chaque service. Nous vous recommandons de terminer la démonstration sur un cluster sur lequel vous n'exécutez pas de charges de travail de production.

Avant de créer des politiques réseau, tous les services peuvent communiquer de façon bidirectionnelle. Une fois que vous avez appliqué les politiques réseau, vous constatez que le client peut uniquement communiquer avec le service frontend et que le service backend n'accepte que le trafic du frontend.

1. Appliquez les services frontal, backend et d'interface utilisateur de gestion :

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
```

2. Afficher tous les pods du cluster.

```
kubectl get pods -A
```

L'exemple qui suit illustre un résultat.

Dans votre sortie, vous devriez voir des pods dans les espaces de noms affichés dans la sortie suivante. Le nombre *NAMES* de vos pods et le nombre de pods dans la `READY` colonne sont différents de ceux de la sortie suivante. Ne poursuivez pas tant que vous n'avez pas vu des modules portant des noms similaires et qu'ils `Running` en ont tous un dans la `STATUS` colonne.

```
NAMESPACE         NAME                                        READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                                1/1     Running   0
     5m19s
[...]
management-ui     management-ui-qrb2g                         1/1     Running   0
     5m24s
stars             backend-sz87q                               1/1     Running   0
     5m23s
stars             frontend-cscnf                              1/1     Running   0
     5m21s
[...]
```

3. Pour vous connecter à l'interface utilisateur de gestion, connectez-vous à l'`EXTERNAL-IP` du service qui s'exécute sur votre cluster :

```
kubectl get service/management-ui -n management-ui
```

4. Ouvrez le navigateur à l'emplacement indiqué à l'étape précédente. Vous devez voir l'interface utilisateur de gestion. Le nœud C est le service client, le nœud F est le service frontal et le nœud B est le service backend. Chaque nœud a un accès en communication complet à tous les autres nœuds, comme indiqué par les lignes colorées en gras.

5. Appliquez les stratégies réseau suivantes tant dans les espaces de noms `stars` que `client` pour isoler les services les uns des autres :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

Vous pouvez utiliser les commandes suivantes pour appliquer la stratégie aux deux espaces de noms :

```
kubectl apply -n stars -f https://raw.githubusercontent.com/aws-samples/eks-workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

```
kubectl apply -n client -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. Actualisez votre navigateur. Vous constatez que l'interface utilisateur de gestion ne peut plus atteindre aucun des nœuds, ils n'apparaissent donc pas dans l'interface utilisateur.
7. Appliquez les stratégies réseau différentes suivantes pour permettre à l'interface utilisateur de gestion d'accéder aux services. Appliquez cette stratégie pour autoriser l'interface utilisateur :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

Appliquez cette stratégie pour autoriser le client :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

Vous pouvez utiliser les commandes suivantes pour appliquer les deux stratégies :

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. Actualisez votre navigateur. Vous pouvez alors constater que l'interface utilisateur graphique de gestion peut à nouveau atteindre les nœuds, mais que les nœuds ne peuvent pas communiquer entre eux.

9. Appliquez la politique réseau suivante pour autoriser le trafic du service frontal au service backend :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
```

```
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

10 Actualisez votre navigateur. Vous voyez que le frontend peut communiquer avec le backend.

11 Appliquez la stratégie réseau suivante pour autoriser le trafic du client vers le service frontend :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
```

```
        - protocol: TCP
          port: 80
```

12. Actualisez votre navigateur. Vous voyez que le client peut communiquer avec le service frontend. Le service frontend peut toujours communiquer avec le service backend.

13. (Facultatif) Une fois que vous avez terminé avec la démonstration, vous pouvez supprimer ses ressources.

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
```

Même après la suppression des ressources, il peut toujours y avoir des points de terminaison de stratégie réseau sur les nœuds qui peuvent interférer de manière inattendue avec la mise en réseau dans votre cluster. La seule façon sûre de supprimer ces règles est de redémarrer les nœuds ou de terminer tous les nœuds et de les recycler. Pour mettre fin à tous les nœuds, définissez le nombre souhaité du groupe Auto Scaling sur 0, puis sauvegardez jusqu'au nombre souhaité, ou terminez simplement les nœuds.

Déployez des pods dans des sous-réseaux alternatifs avec un réseau personnalisé

S'applique à : les nœuds Linux `IPv4` Fargate, les nœuds Linux avec des instances Amazon EC2

Par défaut, lorsque le plug-in Amazon VPC CNI pour Kubernetes crée des interfaces réseau élastiques secondaires (interfaces réseau) pour votre EC2 nœud Amazon, il les crée dans le même sous-réseau que l'interface réseau principale du nœud. Il associe également les mêmes groupes de sécurité à l'interface réseau secondaire, qui sont associés à l'interface réseau principale. Pour une ou plusieurs des raisons suivantes, vous voudrez peut-être que le plugin crée des interfaces réseau secondaires dans un sous-réseau différent ou associe différents groupes de sécurité aux interfaces réseau secondaires, ou les deux :

- Le nombre d'`IPv4`adresses disponibles dans le sous-réseau dans lequel se trouve l'interface réseau principale est limité. Cela peut limiter le nombre de pods que vous pouvez créer dans le sous-réseau. En utilisant un sous-réseau différent pour les interfaces réseau secondaires, vous pouvez augmenter le nombre d'`IPv4`adresses disponibles pour les pods.
- Pour des raisons de sécurité, vos pods devront peut-être utiliser un sous-réseau ou des groupes de sécurité différents de ceux de l'interface réseau principale du nœud.
- Les nœuds sont configurés dans des sous-réseaux publics et vous souhaitez placer les pods dans des sous-réseaux privés. La table de routage associée à un sous-réseau public comprend une route vers une passerelle Internet. La table de routage associée à un sous-réseau privé n'inclut pas de route vers une passerelle Internet.

> **Tip**
>
> Vous pouvez également ajouter un sous-réseau nouveau ou existant directement à votre cluster Amazon EKS, sans utiliser de réseau personnalisé. Pour de plus amples informations, veuillez consulter the section called "Ajouter un sous-réseau VPC existant à un cluster Amazon EKS depuis la console de gestion".

Considérations

Voici quelques considérations relatives à l'utilisation de cette fonctionnalité.

- Lorsque la mise en réseau personnalisée est activée, aucune adresse IP attribuée à l'interface réseau principale n'est attribuée aux Pods. Seules les adresses IP des interfaces réseau secondaires sont attribuées aux Pods.
- Si votre cluster utilise la `IPv6` famille, vous ne pouvez pas utiliser le réseau personnalisé.

- Si vous prévoyez d'utiliser une mise en réseau personnalisée uniquement pour pallier à l'épuisement des adresses `IPv4`, vous pouvez plutôt créer un cluster à l'aide de la famille `IPv6`. Pour de plus amples informations, veuillez consulter the section called "IPv6".
- Même si les pods déployés sur des sous-réseaux spécifiés pour les interfaces réseau secondaires peuvent utiliser des sous-réseaux et des groupes de sécurité différents de ceux de l'interface réseau principale du nœud, les sous-réseaux et les groupes de sécurité doivent se trouver dans le même VPC que le nœud.
- Pour Fargate, les sous-réseaux sont contrôlés via le profil Fargate. Pour de plus amples informations, veuillez consulter the section called "Définir des profils".

Personnalisation de l'interface réseau secondaire dans les nœuds Amazon EKS

Avant de commencer le didacticiel, procédez comme suit :

- Passez en revue les considérations
- Connaissance de la façon dont le plug-in Amazon VPC CNI pour Kubernetes crée des interfaces réseau secondaires et attribue des adresses IP aux pods. Pour de plus amples informations, veuillez consulter Allocation ENI sur GitHub.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Nous vous recommandons de terminer les étapes de cette rubrique dans un shell Bash. Si vous n'utilisez pas de shell Bash, certaines commandes de script telles que les caractères de continuation de ligne et la façon dont les variables sont définies et utilisées nécessitent un ajustement de votre shell. En outre, les règles de votre shell en matière de guillemets peuvent

être différentes. Pour plus d'informations, consultez la section Utilisation de guillemets avec des chaînes dans la AWS CLI du Guide de l'utilisateur de l'interface de ligne de AWS commande.

Pour ce didacticiel, nous vous recommandons d'utiliser les exemples de valeurs, sauf lorsqu'il est indiqué de les remplacer. Vous pouvez remplacer n'importe quel exemple de valeur lorsque vous terminez les étapes d'un cluster de production. Nous vous recommandons de terminer toutes les étapes dans le même terminal. Cela est dû au fait que les variables sont définies et utilisées tout au long des étapes et n'existeront pas dans les différents terminaux.

Les commandes de cette rubrique sont mises en forme selon les conventions répertoriées dans la section Exemples d'utilisation de la AWS CLI. Si vous exécutez des commandes depuis la ligne de commande sur des ressources situées dans une AWS région différente de la AWS région par défaut définie dans le profil AWS CLI que vous utilisez, vous `--region us-west-2` devez ajouter des commandes en les `us-west-2` remplaçant par votre AWS région.

Lorsque vous souhaitez déployer une mise en réseau personnalisée sur votre cluster de production, passez à the section called "Étape 2 : configuration de votre VPC".

Étape 1 : créer un VPC test et un cluster

Les procédures suivantes vous aident à créer un VPC test et un cluster et à configurer une mise en réseau personnalisée pour ce cluster. Nous ne recommandons pas d'utiliser le cluster de test pour les charges de travail de production, car plusieurs fonctionnalités indépendantes que vous pourriez utiliser sur votre cluster de production ne sont pas abordées dans cette rubrique. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster ".

1. Exécutez la commande suivante pour définir la `account_id` variable.

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Créez un VPC.

   a. Si vous effectuez un déploiement sur un système de test, créez un VPC à l'aide d'un modèle Amazon EKS AWS CloudFormation .

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
```

```
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

b. La création de la AWS CloudFormation pile prend quelques minutes. Pour vérifier l'état de déploiement de la pile, exécutez la commande suivante.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

Ne passez pas à l'étape suivante tant que le résultat de la commande n'est pas`CREATE_COMPLETE`.

c. Définissez les variables avec les valeurs du sous-réseau privé IDs créé par le modèle.

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

d. Définissez des variables avec les zones de disponibilité des sous-réseaux récupérés lors de l'étape précédente.

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. Créez un rôle IAM de cluster.

a. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Créez le rôle IAM de cluster Amazon EKS. Si nécessaire, faites précéder `eks-cluster-role-trust-policy.json` par le chemin d'accès sur votre ordinateur sur lequel vous avez enregistré le fichier lors de l'étape précédente. La commande associe la politique d'approbation que vous avez créée lors de l'étape précédente à ce rôle. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-
role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. Associez la politique gérée par Amazon EKS nommée Amazon EKSCluster Policy au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Créez un cluster Amazon EKS et configurez votre appareil pour qu'il communique avec lui.

a. Créer un cluster.

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws: iam::$account_id:role/
myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds="$subnet_id_1","$subnet_id_2"
```

> **Note**
>
> Vous pouvez recevoir un message d'erreur indiquant que l'une des zones de disponibilité de votre demande n'a pas la capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises

en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

b. La création du cluster prend quelques minutes. Pour vérifier l'état de déploiement du cluster, exécutez la commande suivante.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

Ne passez pas à l'étape suivante tant que le résultat de la commande n'est pas`"ACTIVE"`.

c. Configurez `kubectl` pour communiquer avec votre cluster.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Étape 2 : configuration de votre VPC

Dans ce didacticiel, vous devez disposer du VPC créé dans the section called "Étape 1 : créer un VPC test et un cluster". Pour un cluster de production, ajustez les étapes en conséquence pour votre VPC en remplaçant tous les exemples de valeur par vos propres valeurs.

1. Vérifiez que le plugin Amazon VPC CNI pour Kubernetes actuellement installé est bien la dernière version. Pour déterminer la version la plus récente du type de module complémentaire Amazon EKS et mettre à jour votre version en conséquence, consultez the section called "Mettre à jour un module complémentaire". Pour déterminer la version la plus récente du type de module complémentaire autogéré et mettre à jour votre version en conséquence, consultez the section called "CNI Amazon VPC".
2. Récupérez l'ID du VPC de votre cluster et stockez-le dans une variable pour une utilisation ultérieure.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. Associez un bloc CIDR (Classless Inter-Domain Routing) supplémentaire au VPC de votre cluster. Le bloc d'adresse CIDR ne peut pas se chevaucher avec les blocs d'adresse CIDR associés existants.

   a. Affichez les blocs CIDR actuels associés à votre VPC.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

L'exemple qui suit illustre un résultat.

```
----------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
+-----------------+--------------+
```

b. Associez un bloc CIDR supplémentaire à votre VPC. Remplacez la valeur du bloc CIDR dans la commande suivante. Pour plus d'informations, consultez Associer des blocs IPv4 CIDR supplémentaires à votre VPC dans le guide de l'utilisateur Amazon VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

c. Vérifiez que le nouveau bloc est associé.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

L'exemple qui suit illustre un résultat.

```
----------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

Ne passez pas à l'étape suivante tant que votre nouveau bloc CIDR ne l'`State`est `associated` pas.

4. Créez autant de sous-réseaux que vous souhaitez utiliser dans chaque zone de disponibilité dans laquelle se trouvent vos sous-réseaux existants. Spécifiez un bloc d'adresse CIDR inclus dans le bloc d'adresse CIDR que vous avez associé à votre VPC lors d'une étape précédente.

   a. Créez de nouveaux sous-réseaux. Remplacez les valeurs des blocs CIDR dans la commande suivante. Les sous-réseaux doivent être créés dans un bloc CIDR VPC différent de celui de vos sous-réseaux existants, mais dans les mêmes zones de disponibilité que vos sous-réseaux existants. Dans cet exemple, un sous-réseau est créé dans le nouveau bloc CIDR de chaque zone de disponibilité dans laquelle les sous-réseaux privés actuels existent. Les IDs sous-réseaux créés sont stockés dans des variables à utiliser lors des étapes ultérieures. Les valeurs `Name` correspondent aux valeurs affectées aux sous-réseaux créés à l'aide du modèle Amazon EKS VPC lors d'une étape précédente. Les noms ne sont pas obligatoires. Vous pouvez utiliser des noms différents.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1
 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2
 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> Par défaut, vos nouveaux sous-réseaux sont implicitement associés à la table de routage principale de votre VPC. Cette table de routage permet la communication entre toutes les ressources déployées dans le VPC. Toutefois, il n'autorise pas la communication avec les ressources dont les adresses IP sont situées en dehors des blocs CIDR associés à votre VPC. Vous pouvez associer votre propre table de routage à vos sous-réseaux pour modifier ce comportement. Pour plus d'informations, consultez Tables de routage de sous-réseau dans le Guide de l'utilisateur Amazon VPC.

   b. Affichez les sous-réseaux actuels dans votre VPC.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
```

```
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------------------------------------
|                           DescribeSubnets                           |
+------------------+--------------------+----------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId           |
+------------------+--------------------+----------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1        |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2        |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3        |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4        |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5        |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6        |
+------------------+--------------------+----------------------------+
```

Vous pouvez voir que les sous-réseaux se trouvant dans le bloc CIDR `192.168.1.0` que vous avez créé se trouvent dans les mêmes zones de disponibilité que les sous-réseaux dans le bloc d'adresse CIDR `192.168.0.0`.

Étape 3 : configuration des ressources Kubernetes

1. Définissez la variable d'`AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG`environnement sur `true` dans le `aws-node` DaemonSet.

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. Récupérez l'ID de votre groupe de sécurité de cluster et stockez-le dans une variable pour une utilisation ultérieure. Amazon EKS crée automatiquement ce groupe de sécurité lorsque vous créez votre cluster.

```
cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-
cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Créez une ressource `ENIConfig` personnalisée pour chaque sous-réseau dans lequel vous souhaitez déployer des pods.

a. Créez un fichier unique pour chaque configuration d'interface réseau.

Les commandes suivantes créent des fichiers `ENIConfig` distincts pour les deux sous-réseaux créés à l'étape précédente. La valeur pour `name` doit être unique. Le nom est le même que la zone de disponibilité dans laquelle se trouve le sous-réseau. Le groupe de sécurité de cluster est affecté au `ENIConfig`.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

Pour un cluster de production, vous pouvez apporter les modifications suivantes aux commandes précédentes :

- Remplacez $cluster_security_group_id par l'ID d'un groupe de sécurité existant que vous souhaitez utiliser pour chacun d'eux. `ENIConfig`
- Nous vous recommandons de donner à votre zone de disponibilité `ENIConfigs` le même nom que celui `ENIConfig` pour lequel vous allez l'utiliser, dans la mesure du possible. Vous devrez peut-être utiliser pour vos `ENIConfigs` des noms différents de ceux des zones de disponibilité pour diverses raisons. Par exemple, si vous avez plus de deux sous-réseaux dans la même zone de disponibilité et que vous souhaitez les utiliser tous les deux avec une mise en réseau personnalisée, vous avez besoin de plusieurs `ENIConfigs` pour la

même zone de disponibilité. Comme chacun `ENIConfig` nécessite un nom unique, vous ne pouvez pas nommer plus d'un d'entre eux `ENIConfigs` en utilisant le nom de la zone de disponibilité.

Si vos `ENIConfig` noms ne sont pas tous identiques à ceux des zones de disponibilité, remplacez $az_1 et $az_2 par vos propres noms dans les commandes précédentes et annotez vos nœuds avec les derniers noms de ce didacticiel. ENIConfig

> **Note**
>
> Si vous ne spécifiez pas de groupe de sécurité valide à utiliser avec un cluster de production et que vous utilisez :

- version `1.8.0` ou ultérieure du plugin Amazon VPC CNI pour Kubernetes, puis les groupes de sécurité associés à l'interface elastic network principale du nœud sont utilisés.
- une version du plugin Amazon VPC CNI pour Kubernetes antérieure à`1.8.0`, puis le groupe de sécurité par défaut pour le VPC est attribué aux interfaces réseau secondaires.

> **Important**
>
> - `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` est un paramètre par défaut dans la configuration du plugin CNI Amazon VPC pour Kubernetes. Si vous utilisez le paramètre par défaut, le trafic destiné aux adresses IP ne figurant pas dans l'un des blocs CIDR associés à votre VPC utilise les groupes de sécurité et les sous-réseaux de l'interface réseau principale de votre nœud. Les sous-réseaux et les groupes de sécurité définis dans votre `ENIConfigs` répertoire et utilisés pour créer des interfaces réseau secondaires ne sont pas utilisés pour ce trafic. Pour plus d'informations sur ce paramètre, consultez the section called "Trafic sortant".
> - Si vous utilisez également des groupes de sécurité pour les pods, le groupe de sécurité spécifié dans a `SecurityGroupPolicy` est utilisé à la place du groupe de sécurité spécifié dans le`ENIConfigs`. Pour de plus amples informations, veuillez consulter the section called "Groupes de sécurité pour Pods".

b. Appliquez chaque fichier de ressource personnalisé que vous avez créé à votre cluster à l'aide des commandes suivantes.

```
kubectl apply -f $az_1.yaml
```

```
kubectl apply -f $az_2.yaml
```

4. Confirmez que vos `ENIConfigs` ont été créés.

```
kubectl get ENIConfigs
```

L'exemple qui suit illustre un résultat.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. Si vous activez une mise en réseau personnalisée sur un cluster de production et que vous `ENIConfigs` lui avez donné un nom autre que la zone de disponibilité pour laquelle vous les utilisez, passez à l'étape suivante pour déployer des EC2 nœuds Amazon.

   Permettez à Kubernetes d'appliquer automatiquement la zone `ENIConfig` de disponibilité à tous les nouveaux EC2 nœuds Amazon créés dans votre cluster.

   a. Pour le cluster test de ce didacticiel, passez à l'étape suivante.

      Pour un cluster de production, vérifiez si une annotation contenant la clé de la variable `k8s.amazonaws.com/eniConfig` d' `ENI_CONFIG_ANNOTATION_DEF` environnement existe dans la spécification du conteneur pour le `aws-node` DaemonSet.

```
kubectl describe daemonset aws-node -n kube-system | grep
 ENI_CONFIG_ANNOTATION_DEF
```

      Si la sortie est renvoyée, l'annotation existe. Si aucune sortie n'est renvoyée, la variable n'est pas définie. Pour un cluster de production, vous pouvez utiliser ce paramètre ou le paramètre de l'étape suivante. Si vous utilisez ce paramètre, il remplace le paramètre de l'étape suivante. Dans ce didacticiel, le paramètre de l'étape suivante est utilisé.

   b. Mettez `aws-node` DaemonSet à jour votre `ENIConfig` pour appliquer automatiquement la zone de disponibilité à tous les nouveaux EC2 nœuds Amazon créés dans votre cluster.

```
kubectl set env daemonset aws-node -n kube-system
 ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

Étape 4 : Déployer des EC2 nœuds Amazon

1. Créez un rôle IAM de nœud.

   a. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

   b. Créez un rôle IAM et stockez le nom de ressource Amazon (ARN) renvoyé dans une variable pour une utilisation ultérieure.

```
node_role_arn=$(aws iam create-role --role-name myCustomNetworkingNodeRole --
assume-role-policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

   c. Attachez trois politiques gérées IAM requises au rôle IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name myCustomNetworkingNodeRole
```

> **Important**
>
> Pour simplifier ce didacticiel, la politique Amazoneks_CNI_Policy est attachée au rôle IAM du nœud. Dans un cluster de production, nous recommandons toutefois d'associer la politique à un rôle IAM distinct qui est utilisé uniquement avec le plug-in Amazon VPC CNI pour Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

2. Créez l'un des types de groupes de nœuds suivants. Pour déterminer le type d'instance que vous souhaitez déployer, consultez the section called "Types d' EC2 instances Amazon". Pour ce didacticiel, terminez l'option Gérée, Sans modèle de lancement ou avec un modèle de lancement sans ID d'AMI spécifié. Si vous comptez utiliser le groupe de nœuds pour les charges de travail de production, nous vous recommandons de vous familiariser avec toutes les options de groupe de nœuds gérés et de groupes de nœuds autogérés avant de déployer le groupe de nœuds.
   - Géré : déployez votre groupe de nœuds à l'aide de l'une des options suivantes :
     - Sans modèle de lancement ou avec un modèle de lancement sans ID d'AMI spécifié : exécutez la commande suivante. Dans le cadre de ce didacticiel, utilisez les exemples de valeurs. Pour un groupe de nœuds de production, remplacez tous les exemples de valeurs par les vôtres. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

```
aws eks create-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup \
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
 $node_role_arn
```

     - Avec un modèle de lancement avec un ID AMI spécifié

       A. Déterminez le nombre maximum de pods recommandé par Amazon EKS pour vos nœuds. Suivez les instructions de la section Nombre maximal de pods recommandé EC2 par Amazon EKS pour chaque type d'instance Amazon, `--cni-custom-networking-enabled` en complément de l'étape 3 de cette rubrique. Notez la sortie pour l'utiliser lors de l'étape suivante.

       B. Dans votre modèle de lancement, spécifiez un ID d'AMI optimisé pour Amazon EKS ou une AMI personnalisée créée à partir de l'AMI optimisée pour Amazon EKS, puis déployez le groupe de nœuds avec un modèle de lancement et fournissez les données

utilisateur suivantes dans le modèle de lancement. Ces données utilisateur transmettent des arguments dans le fichier `bootstrap.sh`. Pour plus d'informations sur le fichier d'amorçage, consultez bootstrap.sh sur GitHub. Vous pouvez remplacer `20` par la valeur de l'étape précédente (recommandé) ou par votre propre valeur.

```
/etc/eks/bootstrap.sh my-custom-networking-cluster --use-max-pods false --
kubelet-extra-args '--max-pods=20'
```

Si vous avez créé une AMI personnalisée qui n'est pas basée sur l'AMI optimisée pour Amazon EKS, vous devez créer vous-même la configuration sur mesure.

- Autogéré

  i. Déterminez le nombre maximum de pods recommandé par Amazon EKS pour vos nœuds. Suivez les instructions de la section the section called "Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon", en ajoutant `--cni-custom-networking-enabled` à l'étape 3 de cette rubrique. Notez la sortie pour l'utiliser lors de l'étape suivante.

  ii. Déployez le groupe de nœuds à l'aide des instructions contenues dans the section called "Amazon Linux". Spécifiez le texte suivant pour le BootstrapArgumentsparamètre. Vous pouvez remplacer `20` par la valeur de l'étape précédente (recommandé) ou par votre propre valeur.

```
--use-max-pods false --kubelet-extra-args '--max-pods=20'
```

> **Note**
>
> Si vous souhaitez que les nœuds d'un cluster de production prennent en charge un nombre nettement plus élevé de pods, the section called "Nombre maximum de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon" réexécutez le script. Ajoutez également l'option `--cni-prefix-delegation-enabled` à la commande. Par exemple, `110` est renvoyé pour un type d'instance `m5.large`. Pour obtenir des instructions sur la façon d'activer cette capacité, consultez the section called "Augmenter le nombre d'adresses IP". Vous pouvez utiliser cette capacité avec une mise en réseau personnalisée.

3. La création d'un groupe de nœuds dure plusieurs minutes. Vous pouvez vérifier l'état de la création d'un groupe de nœuds géré à l'aide de la commande suivante.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

Ne passez pas à l'étape suivante tant que le résultat renvoyé ne l'est pas`ACTIVE`.

4. Pour le didacticiel, vous pouvez ignorer cette étape.

   Pour un cluster de production, si vous n'avez pas donné `ENIConfigs` le même nom que la zone de disponibilité pour laquelle vous l'utilisez, vous devez annoter vos nœuds avec le `ENIConfig` nom qui doit être utilisé avec le nœud. Cette étape n'est pas nécessaire si vous n'avez qu'un seul sous-réseau dans chaque zone de disponibilité et que vous les avez nommées `ENIConfigs` avec les mêmes noms que vos zones de disponibilité. Cela est dû au fait que le plug-in Amazon VPC CNI pour Kubernetes associe automatiquement le nœud correct `ENIConfig` au nœud pour vous lorsque vous l'avez activé lors d'une étape précédente.

   a. Obtenez la liste des nœuds de votre cluster.

   ```
   kubectl get nodes
   ```

   L'exemple qui suit illustre un résultat.

   ```
   NAME                                          STATUS   ROLES    AGE     VERSION
   ip-192-168-0-126.us-west-2.compute.internal   Ready    <none>   8m49s   v1.22.9-
   eks-810597c
   ip-192-168-0-92.us-west-2.compute.internal    Ready    <none>   8m34s   v1.22.9-
   eks-810597c
   ```

   b. Déterminez dans quelle zone de disponibilité se trouve chaque nœud. Exécutez la commande suivante pour chaque nœud renvoyé à l'étape précédente, en remplaçant les adresses IP en fonction de la sortie précédente.

   ```
   aws ec2 describe-instances --filters Name=network-interface.private-dns-
   name,Values=ip-192-168-0-126.us-west-2.compute.internal \
   --query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone,
    SubnetId: SubnetId}'
   ```

   L'exemple qui suit illustre un résultat.

   ```
   [
       {
   ```

```
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. Annotez chaque nœud avec le `ENIConfig` que vous avez créé pour l'ID de sous-réseau et la zone de disponibilité. Vous ne pouvez annoter un nœud qu'avec un seul `ENIConfig`, bien que plusieurs nœuds puissent être annotés avec le même `ENIConfig`. Remplacez les exemples de valeurs par les vôtres.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

5. Si vous disposiez de nœuds dans un cluster de production avec des pods en cours d'exécution avant de passer à la fonctionnalité de mise en réseau personnalisée, effectuez les tâches suivantes :

   a. Assurez-vous que vous disposez de nœuds disponibles utilisant la fonctionnalité de mise en réseau personnalisée.

   b. Bouclez et videz les nœuds pour éteindre les Pods avec élégance. Pour plus d'informations, consultez Drainer un nœud en toute sécurité dans la documentation Kubernetes.

   c. Mettez fin aux nœuds. Si les nœuds se trouvent dans un groupe de nœuds géré existant, vous pouvez supprimer le groupe de nœuds. Exécutez la commande suivante.

   ```
   aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
   name my-nodegroup
   ```

   Seuls les nouveaux nœuds enregistrés auprès du label `k8s.amazonaws.com/eniConfig` utilisent la fonctionnalité de réseaux personnalisés.

6. Vérifiez qu'une adresse IP est attribuée aux pods à partir d'un bloc CIDR associé à l'un des sous-réseaux que vous avez créés à l'étape précédente.

   ```
   kubectl get pods -A -o wide
   ```

   L'exemple qui suit illustre un résultat.

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

Vous pouvez constater que les adresses IP attribuées aux pods coredns proviennent du bloc `192.168.1.0` CIDR que vous avez ajouté à votre VPC. Sans mise en réseau personnalisée, des adresses auraient été attribuées à partir du bloc CIDR `192.168.0.0`, car il s'agissait du seul bloc CIDR initialement associé au VPC.

Si un pod en `spec` contient`hostNetwork=true`, l'adresse IP principale du nœud lui est attribuée. Aucune adresse ne lui est attribuée à partir des sous-réseaux que vous avez ajoutés. Par défaut, cette valeur indique `false`. Cette valeur est définie sur `true` pour le `kube-proxy` plugin Amazon VPC CNI pour les pods Kubernetes (`aws-node`) qui s'exécutent sur votre cluster. C'est pourquoi aucune adresse 192.168.1.x n'a été attribuée aux `aws-node` pods `kube-proxy` et au plugin dans la sortie précédente. Pour plus d'informations sur les `hostNetwork` paramètres d'un pod, consultez la section PodSpec v1 core dans la référence de l'API Kubernetes.

Étape 5 : suppression des ressources du didacticiel

Une fois le didacticiel terminé, nous vous recommandons de supprimer les ressources que vous avez créées. Vous pouvez ensuite ajuster les étapes pour activer la mise en réseau personnalisée pour un cluster de production.

1. Si le groupe de nœuds que vous avez créé n'était qu'à des fins de test, supprimez-le.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name
 my-nodegroup
```

2. Même une fois que la sortie de la AWS CLI indique que le cluster est supprimé, le processus de suppression peut ne pas être terminé. Le processus de suppression prend quelques minutes. Vérifiez qu'il est terminé en exécutant la commande suivante.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   Ne continuez pas tant que la sortie renvoyée n'est pas similaire à la sortie suivante.

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

3. Si le groupe de nœuds que vous avez créé n'était qu'à des fins de test, supprimez le rôle IAM du nœud.

   a. Détachez les politiques du rôle.

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
```

   b. Supprimez le rôle.

```
aws iam delete-role --role-name myCustomNetworkingNodeRole
```

4. Supprimez le cluster.

```
aws eks delete-cluster --name my-custom-networking-cluster
```

Vérifiez que le cluster est supprimé avec la commande suivante.

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
 --output text
```

Lorsqu'une sortie similaire à la suivante est renvoyée, le cluster est correctement supprimé.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-custom-networking-cluster.
```

5. Supprimez le rôle IAM de cluster.
   a. Détachez les politiques du rôle.

   ```
   aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
   policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
   ```

   b. Supprimez le rôle.

   ```
   aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
   ```

6. Supprimez les sous-réseaux que vous avez créés à l'étape précédente.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

7. Supprimez le VPC que vous avez créé.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Attribuez davantage d'adresses IP aux nœuds Amazon EKS avec des préfixes

S'applique à : nœuds Linux et Windows dotés d' EC2 instances Amazon

S'applique à : sous-réseaux publics et privés

Chaque EC2 instance Amazon prend en charge un nombre maximum d'interfaces réseau élastiques et un nombre maximum d'adresses IP pouvant être attribuées à chaque interface réseau. Chaque

nœud a besoin d'une adresse IP pour chaque interface réseau. Toutes les autres adresses IP disponibles peuvent être attribuées aux `Pods`. Chaque `Pod` a besoin de sa propre adresse IP. Par conséquent, certains nœuds disposent peut-être de ressources de calcul et de mémoire disponibles, mais ne peuvent pas en accueillir d'autres `Pods` car le nœud n'a plus d'adresses IP à attribuer`Pods`.

Vous pouvez augmenter le nombre d'adresses IP que les nœuds peuvent attribuer `Pods` en attribuant des préfixes IP, plutôt qu'en attribuant des adresses IP secondaires individuelles à vos nœuds. Chaque préfixe inclut plusieurs adresses IP. Si vous ne configurez pas votre cluster pour l'attribution de préfixes IP, votre cluster doit effectuer davantage d'appels à l'interface de programmation d' EC2 applications (API) Amazon pour configurer les interfaces réseau et les adresses IP nécessaires à la connectivité des Pod. À mesure que les clusters atteignent des tailles plus importantes, la fréquence de ces appels d'API peut allonger les délais de lancement des Pod et des instances. Cela entraîne des retards de mise à l'échelle pour répondre à la demande de charges de travail importantes et pointues, et augmente les coûts et les frais de gestion, car vous devez fournir des clusters supplémentaires et répondre aux exigences de mise VPCs à l'échelle. Pour plus d'informations, consultez la section Seuils d'évolutivité de Kubernetes activés. GitHub

Compatibilité avec le plugin Amazon VPC CNI pour les fonctionnalités de Kubernetes

Vous pouvez utiliser des préfixes IP dotés des fonctionnalités suivantes :

- IPv4 Traduction de l'adresse réseau source - Pour plus d'informations, consultezthe section called “Trafic sortant”.
- IPv6 adresses vers les clusters, les pods et les services - Pour plus d'informations, consultezthe section called “IPv6”.
- Restreindre le trafic à l'aide des politiques réseau Kubernetes - Pour plus d'informations, consultez. the section called “Politiques Kubernetes”

La liste suivante fournit des informations sur les paramètres du plug-in Amazon VPC CNI qui s'appliquent. Pour plus d'informations sur chaque paramètre, voir amazon-vpc-cni-k8s on GitHub.

- `WARM_IP_TARGET`
- `MINIMUM_IP_TARGET`
- `WARM_PREFIX_TARGET`

Considérations

Lorsque vous utilisez cette fonctionnalité, tenez compte des points suivants :

- Chaque type d' EC2 instance Amazon prend en charge un nombre maximum de pods. Si votre groupe de nœuds gérés est composé de plusieurs types d'instances, le plus petit nombre maximum de pods pour une instance du cluster est appliqué à tous les nœuds du cluster.
- Par défaut, le nombre maximum de `Pods` que vous pouvez exécuter sur un nœud est de 110, mais vous pouvez modifier ce nombre. Si vous modifiez le numéro et que vous disposez d'un groupe de nœuds géré existant, la prochaine mise à jour d'AMI ou de modèle de lancement de votre groupe de nœuds entraînera la création de nouveaux nœuds avec la valeur modifiée.
- Lorsque vous passez de l'attribution d'adresses IP à l'attribution de préfixes IP, nous vous recommandons de créer de nouveaux groupes de nœuds afin d'augmenter le nombre d'adresses IP disponibles, plutôt que de remplacer progressivement les nœuds existants. L'exécution de pods sur un nœud auquel des adresses IP et des préfixes sont attribués peut entraîner une incohérence dans la capacité d'adresses IP annoncée, ce qui a un impact sur les futures charges de travail du nœud. Pour connaître la méthode recommandée pour effectuer la transition, consultez Remplacer tous les nœuds lors de la migration du mode IP secondaire vers le mode Délégation de préfixe ou vice versa dans le guide des meilleures pratiques Amazon EKS.
- L'étendue du groupe de sécurité se situe au niveau du nœud. Pour plus d'informations, consultez la section Groupe de sécurité.
- Les préfixes IP attribués à une interface réseau prennent en charge une densité de pods élevée par nœud et offrent le meilleur moment de lancement.
- Les préfixes et adresses IP sont associés aux interfaces réseau Amazon EC2 Elastic standard. Les pods nécessitant des groupes de sécurité spécifiques se voient attribuer l'adresse IP principale d'une interface réseau de branche. Vous pouvez mélanger des pods recevant des adresses IP ou des adresses IP provenant de préfixes IP avec des pods recevant des interfaces réseau de succursales sur le même nœud.
- Pour les clusters dotés de nœuds Linux uniquement.
  - Après avoir configuré le module complémentaire pour attribuer des préfixes aux interfaces réseau, vous ne pouvez pas rétrograder votre module complémentaire Amazon VPC CNI pour Kubernetes vers une version inférieure à `1.9.0` (ou`1.10.1`) sans supprimer tous les nœuds de tous les groupes de nœuds de votre cluster.
  - Si vous utilisez également des groupes de sécurité pour les pods, avec `POD_SECURITY_GROUP_ENFORCING_MODE` = `standard` et `AWS_VPC_K8S_CNI_EXTERNALSNAT` =`false`, lorsque vos pods communiquent avec des points

de terminaison extérieurs à votre VPC, les groupes de sécurité du nœud sont utilisés, plutôt que les groupes de sécurité que vous avez attribués à vos pods.

Si vous utilisez également des groupes de sécurité pour les pods, avec `POD_SECURITY_GROUP_ENFORCING_MODE =strict`, lorsque vous `Pods` communiquez avec des points de terminaison extérieurs à votre VPC, `Pod's` les groupes de sécurité sont utilisés.

Augmentez le nombre d'adresses IP disponibles pour votre nœud Amazon EKS

Vous pouvez augmenter le nombre d'adresses IP que les nœuds peuvent attribuer aux pods en attribuant des préfixes IP, plutôt qu'en attribuant des adresses IP secondaires individuelles à vos nœuds.

Prérequis

- Vous avez besoin d'un cluster existant. Pour en déployer un, consultez the section called "Créer un cluster ".
- Les sous-réseaux dans lesquels se trouvent vos nœuds Amazon EKS doivent comporter suffisamment de blocs de routage inter-domaines sans classe (CIDR) contigus `/28` (pour les clusters `IPv4`) ou `/80` (pour les clusters `IPv6`). Vous ne pouvez avoir que des nœuds Linux dans un cluster `IPv6`. L'utilisation de préfixes IP peut échouer si les adresses IP sont dispersées dans le CIDR du sous-réseau. Nous vous recommandons la procédure suivante :
  - Utiliser une réservation CIDR de sous-réseau afin que même si des adresses IP comprises dans la plage réservée sont toujours utilisées, elles ne soient pas réattribuées lors de leur publication. Cela permet de s'assurer que les préfixes sont disponibles pour l'allocation sans segmentation.
  - Utilisez de nouveaux sous-réseaux spécifiquement utilisés pour exécuter les charges de travail auxquelles les préfixes IP sont attribués. Les charges de travail Windows et Linux peuvent s'exécuter dans le même sous-réseau lors de l'attribution de préfixes IP.
- Pour attribuer des préfixes IP à vos nœuds, ceux-ci doivent être basés sur AWS Nitro. Les instances qui ne sont pas basées sur Nitro continuent d'allouer des adresses IP secondaires individuelles, mais ont un nombre d'adresses IP à attribuer aux pods nettement inférieur à celui des instances basées sur Nitro.
- Pour les clusters dotés de nœuds Linux uniquement : si votre cluster est configuré pour la `IPv4` famille, vous devez avoir installé une version `1.9.0` ou une version ultérieure du module complémentaire Amazon VPC CNI pour Kubernetes. Vous pouvez vérifier votre version actuelle à l'aide de la commande suivante.

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

Si votre cluster est configuré pour la famille `IPv6`, la version `1.10.1` du module complémentaire doit être installée. Si la version de votre plugin est antérieure aux versions requises, vous devez la mettre à jour. Pour plus d'informations, consultez les sections de mise à jour d'Assign IPs to Pods with the Amazon VPC CNI.

- Pour les clusters dotés de nœuds Windows uniquement
  - La version de votre cluster et de sa plateforme doit être égale ou ultérieure aux versions indiquées dans le tableau suivant. Pour mettre à jour la version de votre cluster, consultez the section called "Mettre à jour la version de Kubernetes". Si votre cluster ne dispose pas de la version minimale de plate-forme, vous ne pouvez pas attribuer de préfixes IP à vos nœuds tant qu'Amazon EKS n'a pas mis à jour la version de votre plate-forme.

Version de Kubernetes	Version de la plateforme
`1.27`	`eks.3`
`1.26`	`eks.4`

    Vous pouvez vérifier votre version actuelle de Kubernetes et de votre plateforme *`my-cluster`* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :. `aws eks describe-cluster --name` *`my-cluster`* `--query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'`
  - Le support Windows doit être activé pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Activer le support Windows".

Attribuer des préfixes d'adresses IP aux nœuds

Configurez votre cluster pour attribuer des préfixes d'adresses IP aux nœuds. Effectuez la procédure correspondant au système d'exploitation de votre nœud.

Linux

1. Activez le paramètre pour attribuer des préfixes aux interfaces réseau pour le CNI Amazon DaemonSet VPC. Lorsque vous déployez un cluster, une version `1.10.1` ou une version ultérieure du plug-in Amazon VPC CNI pour Kubernetes est déployée avec celui-ci. Si vous avez créé le cluster avec la famille `IPv6`, ce paramètre a été réglé sur `true` par défaut. Si vous avez créé le cluster avec la famille `IPv4`, ce paramètre a été réglé sur `false` par défaut.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
```

> **Important**
>
> Même si votre sous-réseau possède des adresses IP disponibles, si aucun `/28` bloc contigu n'est disponible sur le sous-réseau, vous verrez l'erreur suivante s'afficher dans le plug-in Amazon VPC CNI pour les journaux Kubernetes.
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
> blocks to satisfy the request
> ```
>
> Cela peut se produire en raison de la fragmentation des adresses IP secondaires existantes réparties sur un sous-réseau. Pour résoudre cette erreur, créez un nouveau sous-réseau et lancez-y des pods, ou utilisez une réservation CIDR du EC2 sous-réseau Amazon pour réserver de l'espace au sein d'un sous-réseau à utiliser avec l'attribution de préfixes. Pour plus d'informations, consultez la section Réservations CIDR de sous-réseau dans le Guide de l'utilisateur Amazon VPC.

2. Si vous envisagez de déployer un groupe de nœuds gérés sans modèle de lancement, ou avec un modèle de lancement dans lequel vous n'avez pas spécifié d'ID AMI, et que vous utilisez une version du plugin Amazon VPC CNI pour Kubernetes égale ou ultérieure aux versions répertoriées dans les prérequis, passez à l'étape suivante. Les groupes de nœuds gérés calculent automatiquement le nombre maximum de pods pour vous.

   Si vous déployez un groupe de nœuds autogéré ou un groupe de nœuds gérés avec un modèle de lancement dans lequel vous avez spécifié un ID AMI, vous devez déterminer le nombre maximal de pods recommandé par Amazon EKS pour vos nœuds. Suivez les instructions de la section Nombre maximal de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon, en ajoutant `--cni-prefix-delegation-enabled` à l'étape 3. Notez la sortie pour une utilisation ultérieure.

> **Important**
>
> Les groupes de nœuds gérés appliquent un nombre maximal sur la valeur de `maxPods`. Pour les instances avec moins de 30 v, CPUs le nombre maximum est de 110 et pour toutes les autres instances, le nombre maximum est de 250. Ce nombre maximal est appliqué que la délégation du préfixe soit activée ou non.

3. Si vous utilisez un cluster configuré pour`IPv6`, passez à l'étape suivante.

   Spécifiez les paramètres dans l'une des options suivantes. Pour déterminer quelle option vous convient le mieux et quelle valeur lui fournir, consultez WARM_PREFIX_TARGET, WARM_IP_TARGET et MINIMUM_IP_TARGET on. GitHub

   Vous pouvez remplacer les *`example values`* par une valeur supérieure à zéro.

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` ou `MINIMUM_IP_TARGET` : si l'une ou l'autre des valeurs est définie, elle remplace toute valeur définie pour `WARM_PREFIX_TARGET`.

     ```
     kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
     ```

     ```
     kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
     ```

4. Créez l'un des types de groupes de nœuds suivants avec au moins un type d'instance Amazon EC2 Nitro Amazon Linux 2. Pour obtenir la liste des types d'instances Nitro, consultez la section Instances créées sur le système Nitro dans le guide de EC2 l'utilisateur Amazon. Cette fonctionnalité n'est pas prise en charge sur Windows. Pour les options qui incluent *`110`*, remplacez-le par la valeur de l'étape 3 (recommandée) ou par votre propre valeur.

   - Autogéré : déployez le groupe de nœuds en suivant les instructions de la section Création de nœuds Amazon Linux autogérés. Spécifiez le texte suivant pour le BootstrapArgumentsparamètre.

     ```
     --use-max-pods false --kubelet-extra-args '--max-pods=110'
     ```

Si vous utilisez `eksctl` pour créer le groupe de nœuds, vous pouvez utiliser la commande suivante.

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node
 110
```

- Géré : déployez votre groupe de nœuds à l'aide de l'une des options suivantes :
  - Sans modèle de lancement ou avec un modèle de lancement sans ID d'AMI spécifié : suivez la procédure décrite dans Créer un groupe de nœuds gérés pour votre cluster. Les groupes de nœuds gérés calculent automatiquement pour vous la valeur `max-pods` recommandée par Amazon EKS.
  - Avec un modèle de lancement avec un ID d'AMI spécifié : dans votre modèle de lancement, spécifiez un ID d'AMI optimisé pour Amazon EKS ou une AMI personnalisée créée à partir de l'AMI optimisée pour Amazon EKS, puis Déployez le groupe de nœuds avec un modèle de lancement et fournissez les données utilisateur suivantes dans le modèle de lancement. Ces données utilisateur transmettent des arguments dans le fichier `bootstrap.sh`. Pour plus d'informations sur le fichier d'amorçage, consultez bootstrap.sh sur GitHub.

    ```
    /etc/eks/bootstrap.sh my-cluster \
      --use-max-pods false \
      --kubelet-extra-args '--max-pods=110'
    ```

    Si vous utilisez `eksctl` pour créer le groupe de nœuds, vous pouvez utiliser la commande suivante.

    ```
    eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
    ```

    Si vous avez créé une AMI personnalisée qui n'est pas basée sur l'AMI optimisée pour Amazon EKS, vous devez créer vous-même la configuration personnalisée.

> **Note**
>
> Si vous souhaitez également attribuer des adresses IP à des pods à partir d'un sous-réseau différent de celui de l'instance, vous devez activer cette fonctionnalité à cette étape. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée".

Windows

1. Activez l'attribution de préfixes IP.

   a. Ouvrez le `amazon-vpc-cni` `ConfigMap` pour le modifier.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Ajoutez les lignes suivantes à la section `data`.

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. Enregistrez le fichier et fermez l'éditeur.

   d. Confirmez que la ligne a été ajoutée à la `ConfigMap`.

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-
   windows-prefix-delegation}"
   ```

   Si le résultat renvoyé ne l'est pas`true`, il se peut qu'il y ait eu une erreur. Essayez à nouveau d'effectuer l'étape.

   > **Important**
   >
   > Même si votre sous-réseau possède des adresses IP disponibles, si aucun `/28` bloc contigu n'est disponible sur le sous-réseau, vous verrez l'erreur suivante s'afficher dans le plug-in Amazon VPC CNI pour les journaux Kubernetes.
   >
   > ```
   > InsufficientCidrBlocks: The specified subnet does not have enough free cidr
   > blocks to satisfy the request
   > ```
   >
   > Cela peut se produire en raison de la fragmentation des adresses IP secondaires existantes réparties sur un sous-réseau. Pour résoudre cette erreur, créez un nouveau sous-réseau et lancez-y des pods, ou utilisez une réservation CIDR du EC2 sous-réseau Amazon pour réserver de l'espace au sein d'un sous-réseau à utiliser avec l'attribution de préfixes. Pour plus d'informations, consultez la section Réservations CIDR de sous-réseau dans le Guide de l'utilisateur Amazon VPC.

2. (Facultatif) Spécifiez une configuration supplémentaire pour contrôler le comportement de pré-échelonnement et d'échelonnement dynamique de votre cluster. Pour plus d'informations,

consultez la section Options de configuration avec le mode de délégation de préfixes sous Windows activé GitHub.

a. Ouvrez le `amazon-vpc-cni` `ConfigMap` pour le modifier.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

b. Remplacez le *`example values`* par une valeur supérieure à zéro et ajoutez les entrées dont vous avez besoin dans la section `data` du `ConfigMap`. Si vous définissez une valeur pour `warm-ip-target` ou `minimum-ip-target`, la valeur remplace toute valeur définie pour `warm-prefix-target`.

```
  warm-prefix-target: "1"
  warm-ip-target: "5"
  minimum-ip-target: "2"
```

c. Enregistrez le fichier et fermez l'éditeur.

3. Créez des groupes de nœuds Windows avec au moins un type d'instance Amazon EC2 Nitro. Pour obtenir la liste des types d'instances Nitro, consultez la section Instances créées sur le système Nitro dans le guide de EC2 l'utilisateur Amazon. Par défaut, le nombre maximum de pods que vous pouvez déployer sur un nœud est de 110. Si vous souhaitez augmenter ou diminuer ce nombre, indiquez ce qui suit dans les données utilisateur de la configuration d'amorçage. Remplacez *`max-pods-quantity`* par votre valeur maximale de pods.

```
-KubeletExtraArgs '--max-pods=max-pods-quantity'
```

Si vous déployez des groupes de nœuds gérés, cette configuration doit être ajoutée dans le modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Modèles de lancement". Pour plus d'informations sur les paramètres de configuration du script bootstrap Windows, consultezthe section called "Paramètres de configuration du script d'amorçage".

Déterminer le nombre maximum de pods et les adresses IP disponibles

1. Une fois que vos nœuds sont déployés, affichez les nœuds de votre cluster.

```
kubectl get nodes
```

L'exemple qui suit illustre un résultat.

```
NAME                                          STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m   v1.XX.X-
eks-6b7464
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-
eks-6b7464
```

2. Décrivez l'un des nœuds pour déterminer la valeur de `max-pods` pour le nœud et le nombre d'adresses IP disponibles. Remplacez *`192.168.30.193`* par l'adresse `IPv4` dans le nom de l'un de vos nœuds renvoyés dans la sortie précédente.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
PrivateIPv4Address'
```

L'exemple qui suit illustre un résultat.

```
pods:                                  110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

Dans la sortie précédente, `110` c'est le nombre maximum de pods que Kubernetes déploiera sur le nœud, même si des adresses *`144`* IP sont disponibles.

Attribuer des groupes de sécurité à des pods individuels

S'applique à : nœuds Linux dotés d' EC2 instances Amazon

S'applique à : sous-réseaux privés

Les groupes de sécurité pour Pods intègrent les groupes EC2 de sécurité Amazon aux Pods Kubernetes. Vous pouvez utiliser les groupes EC2 de sécurité Amazon pour définir des règles qui autorisent le trafic réseau entrant et sortant vers et depuis les pods que vous déployez vers des nœuds exécutés sur de nombreux types d' EC2 instances Amazon et sur Fargate. Pour une explication détaillée de cette fonctionnalité, consultez le billet de blog Présentation des groupes de sécurité pour Pods.

Compatibilité avec le plugin Amazon VPC CNI pour les fonctionnalités de Kubernetes

Vous pouvez utiliser des groupes de sécurité pour les pods dotés des fonctionnalités suivantes :

- IPv4 Traduction de l'adresse réseau source - Pour plus d'informations, consultezthe section called "Trafic sortant".
- IPv6 adresses vers les clusters, les pods et les services - Pour plus d'informations, consultezthe section called "IPv6".
- Restreindre le trafic à l'aide des politiques réseau Kubernetes - Pour plus d'informations, consultez. the section called "Politiques Kubernetes"

Considérations

Avant de déployer des groupes de sécurité pour Pods, tenez compte des limites et conditions suivantes :

- Les groupes de sécurité pour Pods ne peuvent pas être utilisés avec les nœuds Windows.
- Les groupes de sécurité pour Pods peuvent être utilisés avec des clusters configurés pour la `IPv6` famille contenant des EC2 nœuds Amazon à l'aide de la version 1.16.0 ou ultérieure du plugin Amazon VPC CNI. Vous pouvez utiliser des groupes de sécurité pour les pods avec une `IPv6` famille de configurations de clusters contenant uniquement des nœuds Fargate en utilisant la version 1.7.7 ou ultérieure du plugin Amazon VPC CNI. Pour de plus amples informations, consultez the section called "IPv6".
- Les groupes de sécurité pour Pods sont pris en charge par la plupart des familles d' EC2 instances Amazon basées sur Nitro, mais pas par toutes les générations d'une famille. Par exemple`m5`, la famille et les générations `c5` `r5m6g`,,`c6g`,, et d'`r6g`instance sont prises en charge. Aucun type d'instance de la famille `t` n'est pris en charge. Pour obtenir la liste complète des types d'instances pris en charge, consultez le fichier limits.go sur. GitHub Vos nœuds doivent être l'un des types d'instance répertoriés qui ont `IsTrunkingCompatible: true` dans ce fichier.
- Si vous utilisez un réseau et des groupes de sécurité personnalisés pour les pods ensemble, le groupe de sécurité spécifié par les groupes de sécurité pour les pods est utilisé à la place du groupe de sécurité spécifié dans le`ENIConfig`.
- Si vous utilisez la version `1.10.2` ou une version antérieure du plugin Amazon VPC CNI et que vous incluez le `terminationGracePeriodSeconds` paramètre dans les spécifications de votre Pod, la valeur du paramètre ne peut pas être nulle.
- Si vous utilisez une version `1.10` ou une version antérieure du plugin Amazon VPC CNI, ou une version `1.11` avec `POD_SECURITY_GROUP_ENFORCING_MODE` =, qui est le paramètre par défaut`strict`, les services Kubernetes de type `NodePort` et `LoadBalancer` utilisant des cibles d'instance `externalTrafficPolicy` définies sur `Local` ne sont pas pris en charge avec les

pods auxquels vous attribuez des groupes de sécurité. Pour plus d'informations sur l'utilisation d'un équilibreur de charge avec des cibles d'instance, consultez the section called "Répartition de charge réseau".

- Si vous utilisez une version `1.10` ou une version antérieure du plugin Amazon VPC CNI ou une version `1.11` avec `POD_SECURITY_GROUP_ENFORCING_MODE =strict`, qui est le paramètre par défaut, le NAT source est désactivé pour le trafic sortant des pods auxquels des groupes de sécurité ont été assignés afin que les règles des groupes de sécurité sortants soient appliquées. Pour accéder à Internet, les pods auxquels des groupes de sécurité sont assignés doivent être lancés sur des nœuds déployés dans un sous-réseau privé configuré avec une passerelle ou une instance NAT. Les pods avec des groupes de sécurité affectés déployés sur des sous-réseaux publics ne sont pas en mesure d'accéder à Internet.

  Si vous utilisez une version `1.11` ou une version ultérieure du plugin avec `POD_SECURITY_GROUP_ENFORCING_MODE =standard`, le trafic Pod destiné à l'extérieur du VPC est traduit vers l'adresse IP de l'interface réseau principale de l'instance. Pour ce trafic, les règles des groupes de sécurité de l'interface réseau principale sont utilisées, plutôt que celles des groupes de sécurité du Pod.
- Pour utiliser la politique réseau de Calico avec des pods associés à des groupes de sécurité, vous devez utiliser la version `1.11.0` ou une version ultérieure du plugin Amazon VPC CNI et définir =. `POD_SECURITY_GROUP_ENFORCING_MODE standard` Dans le cas contraire, le flux de trafic à destination et en provenance des pods associés aux groupes de sécurité n'est pas soumis à l'application de la politique réseau de Calico et est limité à l'application des groupes EC2 de sécurité Amazon uniquement. Pour mettre à jour la version de votre plug-in CNI Amazon VPC, consultez the section called "CNI Amazon VPC"
- Les pods exécutés sur des EC2 nœuds Amazon qui utilisent des groupes de sécurité dans des clusters qui NodeLocal DNSCacheles utilisent ne sont pris en charge qu'avec la version `1.11.0` ou une version ultérieure du plugin Amazon VPC CNI et avec =. `POD_SECURITY_GROUP_ENFORCING_MODE standard` Pour mettre à jour la version de votre plug-in CNI Amazon VPC, consultez the section called "CNI Amazon VPC"
- Les groupes de sécurité pour les pods peuvent entraîner une latence de démarrage plus élevée pour les pods présentant un taux de désabonnement élevé. Cela est dû à la limitation du débit dans le contrôleur de ressources.
- L'étendue du groupe EC2 de sécurité est au niveau du POD. Pour plus d'informations, consultez la section Groupe de sécurité.

Si vous définissez `POD_SECURITY_GROUP_ENFORCING_MODE=standard` et`AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, le trafic destiné aux points de terminaison extérieurs au VPC utilise les groupes de sécurité du nœud, et non les groupes de sécurité du Pod.

Configurer le plug-in Amazon VPC CNI pour Kubernetes pour les groupes de sécurité pour Amazon EKS Pods

Si vous utilisez des Pods avec des EC2 instances Amazon, vous devez configurer le plugin Amazon VPC CNI pour Kubernetes pour les groupes de sécurité

Si vous utilisez uniquement des Fargate Pods et que votre cluster ne contient aucun nœud EC2 Amazon, consultez. the section called "SecurityGroupPolicy"

1. Vérifiez la version actuelle de votre plugin Amazon VPC CNI pour Kubernetes à l'aide de la commande suivante :

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.7.6
   ```

   Si la version de votre plug-in Amazon VPC CNI pour Kubernetes est antérieure à`1.7.7`, mettez-le à jour vers la version ou ultérieure. `1.7.7` Pour de plus amples informations, consultez the section called "CNI Amazon VPC".

2. Ajoutez le lien :iam/home#/policies/arn:aws: iam : :aws:policy/Amazon EKSVPCResource Controller [EKSVPCResourceAmazon Controller, type="console "] policy IAM gérée au rôle de cluster associé à votre cluster Amazon EKS. La politique permet au rôle de gérer les interfaces réseau, leurs adresses IP privées, leur attachement et leur détachement vers et depuis les instances réseau.

   a. Récupérez le nom de votre rôle IAM de cluster et stockez-le dans une variable. Remplacez *my-cluster* par le nom de votre cluster.

      ```
      cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
       --output text | cut -d / -f 2)
      ```

b. Attachez la stratégie au rôle.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Activez le module complémentaire Amazon VPC CNI pour gérer les interfaces réseau pour les pods en définissant la `ENABLE_POD_ENI` variable sur dans le. `true` `aws-node` DaemonSet Une fois ce paramètre défini sur`true`, le module complémentaire crée une ressource `cninode` personnalisée pour chaque nœud du cluster. Le contrôleur de ressources VPC crée et attache une interface réseau spéciale appelée interface réseau de tronc avec la description `aws-k8s-trunk-eni`.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> L'interface réseau de tronc est incluse dans le nombre maximal d'interfaces réseau prises en charge par le type d'instance. Pour obtenir la liste du nombre maximal d'interfaces réseau prises en charge par chaque type d'instance, consultez la section Adresses IP par interface réseau et par type d'instance dans le guide de EC2 l'utilisateur Amazon. Si votre nœud a déjà le nombre maximal d'interfaces réseau standard qui lui sont attachées, le contrôleur de ressources VPC réservera un espace. Vous devrez réduire suffisamment le nombre de vos pods en cours d'exécution pour que le contrôleur puisse détacher et supprimer une interface réseau standard, créer l'interface réseau principale et l'associer à l'instance.

4. Vous pouvez voir quels nœuds ont une ressource personnalisée `CNINode` avec la commande suivante. Si le message `No resources found` est renvoyé, attendez plusieurs secondes et réessayez. L'étape précédente nécessite le redémarrage du plugin Amazon VPC CNI pour Kubernetes Pods, ce qui prend plusieurs secondes.

```
kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

Si vous utilisez des versions CNI VPC antérieures à `1.15`, les étiquettes de nœud ont été utilisées à la place de la ressource personnalisée `CNINode`. Vous pouvez voir quels nœuds ont l'étiquette

de nœud `aws-k8s-trunk-eni` définie sur la valeur à l'`true`aide de la commande suivante. Si le message `No resources found` est renvoyé, attendez plusieurs secondes et réessayez. L'étape précédente nécessite le redémarrage du plugin Amazon VPC CNI pour Kubernetes Pods, ce qui prend plusieurs secondes.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
```

Une fois l'interface réseau de liaison créée, des adresses IP secondaires sont attribuées aux pods à partir de l'interface réseau principale ou des interfaces réseau standard. L'interface de tronc est automatiquement supprimée si le nœud est supprimé.

Lorsque vous déployez ultérieurement un groupe de sécurité pour un pod, le contrôleur de ressources VPC crée une interface réseau spéciale appelée interface réseau de branche avec une description `aws-k8s-branch-eni` et y associe les groupes de sécurité. Les interfaces réseau de branche sont créées en plus des interfaces réseau standard et de tronc attachées au nœud.

Si vous utilisez des sondes Liveness ou Readiness, vous devez également désactiver le démultiplexage TCP anticipé, afin de `kubelet` pouvoir se connecter aux pods sur les interfaces réseau des succursales via TCP. Pour désactiver le démultiplexage anticipé du protocole TCP, exécutez la commande suivante :

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}'
```

> **Note**
>
> Si vous utilisez `1.11.0` ou une version ultérieure du plugin Amazon VPC CNI pour Kubernetes et que vous utilisez set `POD_SECURITY_GROUP_ENFORCING_MODE =standard`, comme décrit à l'étape suivante, vous n'avez pas besoin d'exécuter la commande précédente.

5. Si votre cluster utilise `NodeLocal DNSCache` ou souhaitez utiliser la politique réseau Calico avec vos pods dotés de leurs propres groupes de sécurité, ou si vous avez des services Kubernetes de type `NodePort` cible avec un `externalTrafficPolicy` ensemble de pods auxquels vous souhaitez attribuer des groupes de sécurité, vous devez utiliser la version `1.11.0` ou une version ultérieure du module complémentaire Amazon VPC CNI pour Kubernetes, et vous devez activer le paramètre suivant : `LoadBalancer Local`

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

IMPORTANT : Les règles des groupes de sécurité des pods ne s'appliquent pas au trafic entre les pods ou entre les pods et les services, tels que **`kubelet`** ou**`nodeLocalDNS`**, qui se trouvent sur le même nœud. Les pods utilisant différents groupes de sécurité sur le même nœud ne peuvent pas communiquer car ils sont configurés dans différents sous-réseaux et le routage est désactivé entre ces sous-réseaux. Le trafic sortant des Pods vers des adresses situées en dehors du VPC est une adresse réseau traduite en adresse IP de l'interface réseau principale de l'instance (sauf si vous l'avez également `AWS_VPC_K8S_CNI_EXTERNALSNAT=true` définie). Pour ce trafic, les règles des groupes de sécurité de l'interface réseau principale sont utilisées, plutôt que celles des groupes de sécurité du Pod. ** Pour que ce paramètre s'applique aux pods existants, vous devez redémarrer les pods ou les nœuds sur lesquels les pods s'exécutent.

6. Pour savoir comment utiliser une politique de groupe de sécurité pour votre Pod, consultezthe section called "SecurityGroupPolicy".

Utiliser une politique de groupe de sécurité pour un Amazon EKS Pod

Pour utiliser des groupes de sécurité pour les pods, vous devez disposer d'un groupe de sécurité existant. Les étapes suivantes vous montrent comment utiliser la politique de groupe de sécurité pour un Pod. Sauf indication contraire, effectuez toutes les étapes à partir du même terminal, car les variables utilisées dans les étapes suivantes ne sont pas conservées d'un terminal à l'autre.

Si vous possédez un Pod avec EC2 des instances Amazon, vous devez configurer le plugin avant d'utiliser cette procédure. Pour de plus amples informations, veuillez consulter the section called "Configuration".

1. Créez un espace de noms Kubernetes vers lequel déployer les ressources . Vous pouvez le *`my-namespace`* remplacer par le nom de l'espace de noms que vous souhaitez utiliser.

```
kubectl create namespace my-namespace
```

2. Déployez une politique `SecurityGroupPolicy` Amazon EKS sur votre cluster.
   a. Copiez les contenus suivants sur votre appareil. Vous pouvez les *`podSelector`* remplacer par `serviceAccountSelector` si vous préférez sélectionner des pods en fonction des étiquettes des comptes de service. Vous devez spécifier un sélecteur ou l'autre. Un champ vide `podSelector` (exemple :`podSelector: {}`) sélectionne tous les pods de l'espace de

noms. Vous pouvez *my-role* modifier le nom de votre rôle. Un `serviceAccountSelector` vide sélectionne tous les comptes de service dans l'espace de noms. Vous pouvez le *my-security-group-policy* remplacer par un nom pour votre nom `SecurityGroupPolicy` et *my-namespace* par l'espace de noms `SecurityGroupPolicy` dans lequel vous souhaitez le créer.

Vous devez le *my_pod_security_group_id* remplacer par l'ID d'un groupe de sécurité existant. Si vous n'avez pas de groupe de sécurité existant, vous devez en créer un. Pour plus d'informations, consultez les groupes EC2 de sécurité Amazon pour les instances Linux dans le guide de EC2 l'utilisateur Amazon. Vous pouvez spécifier un groupe IDs de sécurité de 1 à 5. Si vous spécifiez plusieurs identifiants, la combinaison de toutes les règles de tous les groupes de sécurité est effective pour les pods sélectionnés.

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

> **Important**
>
> Le ou les groupes de sécurité que vous spécifiez pour vos pods doivent répondre aux critères suivants :
>
> - Ils doivent exister. S'ils n'existent pas, lorsque vous déployez un Pod correspondant au sélecteur, votre Pod reste bloqué dans le processus de création. Si vous décrivez le Pod, vous verrez un message d'erreur similaire au suivant :`An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist.`

- Ils doivent autoriser les communications entrantes depuis le groupe de sécurité appliqué à vos nœuds (pour`kubelet`) via tous les ports pour lesquels vous avez configuré des sondes.
- Ils doivent autoriser les communications sortantes via TCP les UDP ports 53 vers un groupe de sécurité attribué aux Pods (ou aux nœuds sur lesquels les Pods s'exécutent) exécutant CoreDNS. Le groupe de sécurité de vos pods CoreDNS doit autoriser le trafic TCP entrant UDP et le trafic du port 53 en provenance du groupe de sécurité que vous spécifiez.
- Ils doivent disposer des règles d'entrée et de sortie nécessaires pour communiquer avec les autres pods avec lesquels ils doivent communiquer.
- Ils doivent disposer de règles permettant aux pods de communiquer avec le plan de contrôle Kubernetes si vous utilisez le groupe de sécurité avec Fargate. La façon la plus simple de le faire est de spécifier le groupe de sécurité de cluster comme l'un des groupes de sécurité.

Les politiques relatives aux groupes de sécurité ne s'appliquent qu'aux nouveaux pods planifiés. Ils n'affectent pas le fonctionnement des Pods.

b. Déployez la politique.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Déployez un exemple d'application avec une étiquette correspondant à la valeur *my-role* pour *podSelector* que vous avez spécifié à l'étape précédente.

   a. Copiez les contenus suivants sur votre appareil. Remplacez le *example values* par le vôtre, puis exécutez la commande modifiée. Si vous remplacez*my-role*, assurez-vous qu'elle est identique à la valeur que vous avez spécifiée pour le sélecteur à l'étape précédente.

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
```

```
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. Pour déployer l'application, exécutez la commande suivante. Lorsque vous déployez l'application, le plugin Amazon VPC CNI pour Kubernetes correspond à l'`role`étiquette et les groupes de sécurité que vous avez spécifiés à l'étape précédente sont appliqués au Pod.

```
kubectl apply -f sample-application.yaml
```

4. Affichez les pods déployés avec l'exemple d'application. Pour le reste de ce sujet, ce terminal est appelé `TerminalA`.

```
kubectl get pods -n my-namespace -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME                              READY   STATUS    RESTARTS   AGE     IP
  NODE                                                NOMINATED NODE   READINESS GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s   192.168.70.145
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s   192.168.73.207
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> Suivez ces conseils si des pods sont bloqués.
>
> - Si des pods sont bloqués dans `Waiting` cet état, exécutez-les`kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* . Si vous voyez `Insufficient permissions: Unable to create Elastic Network Interface.`, vérifiez que vous avez ajouté la politique IAM au rôle de cluster IAM dans une étape précédente.
> - Si des pods sont bloqués, vérifiez que `Pending` le type d'instance de votre nœud est répertorié dans limits.go et que le produit du nombre maximum d'interfaces réseau de succursales prises en charge par le type d'instance multiplié par le nombre de nœuds de votre groupe de nœuds n'est pas déjà atteint. Par exemple, une instance `m5.large` prend en charge neuf interfaces réseau de branche. Si votre groupe de nœuds comporte cinq nœuds, un maximum de 45 interfaces réseau de branche peut être créé pour le groupe de nœuds. Le 46e pod que vous tentez de déployer restera en `Pending` état jusqu'à ce qu'un autre pod associé à des groupes de sécurité soit supprimé.

Si vous exécutez `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* et recevez un message similaire au message suivant, il peut être ignoré en toute sécurité. Ce message peut s'afficher lorsque le plug-in Amazon VPC CNI pour Kubernetes tente de configurer le réseau hôte et échoue lors de la création de l'interface réseau. Le plugin journalise cet événement jusqu'à ce que l'interface réseau soit créée.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f"
 network for Pod "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace" network:
 add cmd: failed to assign an IP address to container
```

Vous ne pouvez pas dépasser le nombre maximum de pods pouvant être exécutés sur le type d'instance. Pour obtenir la liste du nombre maximal de pods que vous pouvez exécuter sur chaque type d'instance, consultez eni-max-pods.txt on GitHub. Lorsque vous supprimez un pod auquel des groupes de sécurité sont associés, ou que vous supprimez le nœud sur lequel le pod est exécuté, le contrôleur de ressources VPC supprime l'interface réseau de la succursale. Si vous supprimez un cluster contenant des pods en utilisant des pods pour les groupes de sécurité, le contrôleur ne supprime pas les interfaces réseau des succursales. Vous devrez donc les supprimer vous-même. Pour plus d'informations sur la suppression d'interfaces réseau, consultez Supprimer une interface réseau dans le guide de EC2 l'utilisateur Amazon.

5. Dans un terminal séparé, décochez l'un des pods. Pour le reste de ce sujet, ce terminal est appelé `TerminalB`. *`5df6f7687b-4fbjm`*Remplacez-le par l'ID de l'un des pods renvoyés dans votre sortie de l'étape précédente.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. À partir du shell dans `TerminalB`, confirmez que l'exemple d'application fonctionne.

```
curl my-app
```

L'exemple qui suit illustre un résultat.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

Vous avez reçu le résultat car tous les pods exécutant l'application sont associés au groupe de sécurité que vous avez créé. Ce groupe contient une règle qui autorise tout le trafic entre tous les pods auxquels le groupe de sécurité est associé. Le trafic DNS est autorisé à partir de ce groupe

de sécurité vers le groupe de sécurité du cluster associé à vos nœuds. Les nœuds exécutent les pods CoreDNS, dont le nom a été recherché par vos pods.

7. À partir du `TerminalA`, supprimez les règles de groupe de sécurité qui autorisent la communication DNS au groupe de sécurité du cluster de votre groupe de sécurité. Si vous n'avez pas ajouté les règles DNS au groupe de sécurité du cluster lors d'une étape précédente, *`$my_cluster_security_group_id`* remplacez-les par l'ID du groupe de sécurité dans lequel vous avez créé les règles.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. À partir du `TerminalB`, essayez de nouveau d'accéder à l'application.

```
curl my-app
```

L'exemple qui suit illustre un résultat.

```
curl: (6) Could not resolve host: my-app
```

La tentative échoue car le pod n'est plus en mesure d'accéder aux pods CoreDNS auxquels le groupe de sécurité du cluster est associé. Le groupe de sécurité du cluster ne dispose plus des règles de groupe de sécurité qui autorisent les communications DNS depuis le groupe de sécurité associé à votre Pod.

Si vous tentez d'accéder à l'application en utilisant les adresses IP renvoyées pour l'un des pods lors d'une étape précédente, vous recevez toujours une réponse car tous les ports sont autorisés entre les pods auxquels le groupe de sécurité est associé et aucune recherche de nom n'est requise.

9. Une fois l'expérimentation terminée, vous pouvez supprimer les exemples de politique de groupe de sécurité, d'application et de groupe de sécurité que vous avez créés. Exécutez les commandes suivantes à partir du `TerminalA`.

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
```

```
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Associez plusieurs interfaces réseau aux pods

Par défaut, le plug-in Amazon VPC CNI attribue une adresse IP à chaque pod. Cette adresse IP est attachée à une interface elastic network qui gère tout le trafic entrant et sortant du pod. Pour augmenter la bande passante et les performances du débit de paquets par seconde, vous pouvez utiliser la fonction multi-NIC du VPC CNI pour configurer un pod multi-hébergement. Un pod multi-hébergement est un pod Kubernetes unique qui utilise plusieurs interfaces réseau (et plusieurs adresses IP). En exécutant un pod multi-hébergement, vous pouvez répartir le trafic de ses applications sur plusieurs interfaces réseau en utilisant des connexions simultanées. Cela est particulièrement utile pour les cas d'utilisation de l'intelligence artificielle (IA), du Machine Learning (ML) et du calcul haute performance (HPC).

Le schéma suivant montre un module multi-hébergement exécuté sur un nœud de travail avec plusieurs cartes d'interface réseau (NICs) utilisées.

Contexte

Sur Amazon EC2, une interface réseau élastique est un composant réseau logique d'un VPC qui représente une carte réseau virtuelle. Pour de nombreux types d' EC2 instances, les interfaces réseau partagent une seule carte d'interface réseau (NIC) matérielle. Cette carte réseau unique dispose d'une bande passante et d'un débit de paquets par seconde maximum.

Si la fonctionnalité Multi-NIC est activée, le VPC CNI n'attribue pas d'adresses IP en masse, ce qu'il fait par défaut. Au lieu de cela, le VPC CNI attribue une adresse IP à une interface réseau sur chaque carte réseau à la demande lorsqu'un nouveau pod démarre. Ce comportement réduit le taux d'épuisement des adresses IP, qui est accru par l'utilisation de modules multihébergés. Étant donné que le VPC CNI attribue une adresse IP à la demande, le démarrage des pods peut prendre plus de temps sur les instances où la fonctionnalité multi-NIC est activée.

Considérations

- Assurez-vous que votre cluster Kubernetes exécute la version VPC CNI ou ultérieure. `1.20.0` La fonctionnalité multi-NIC n'est disponible que dans la version `1.20.0` du VPC CNI ou ultérieure.
- Activez la variable d'`ENABLE_MULTI_NIC`environnement dans le plugin VPC CNI. Vous pouvez exécuter la commande suivante pour définir la variable et démarrer le déploiement du DaemonSet.
  - `kubectl set env daemonset aws-node -n kube-system ENABLE_MULTI_NIC=true`
- Assurez-vous de créer des nœuds de travail dotés de plusieurs cartes d'interface réseau (NICs). Pour obtenir la liste des EC2 instances dotées de plusieurs cartes d'interface réseau, consultez la section Cartes réseau du guide de EC2 l'utilisateur Amazon.
- Si la fonctionnalité Multi-NIC est activée, le VPC CNI n'attribue pas d'adresses IP en masse, ce qu'il fait par défaut. Étant donné que le VPC CNI attribue une adresse IP à la demande, le démarrage des pods peut prendre plus de temps sur les instances où la fonctionnalité multi-NIC est activée. Pour de plus amples informations, veuillez consulter la section précédente the section called "Contexte".
- Lorsque la fonctionnalité Multi-NIC est activée, les pods ne disposent pas de plusieurs interfaces réseau par défaut. Vous devez configurer chaque charge de travail pour utiliser plusieurs cartes réseau. Ajoutez l'`k8s.amazonaws.com/nicConfig: multi-nic-attachment`annotation aux charges de travail qui doivent comporter plusieurs interfaces réseau.

Considérations relatives à **IPv6**

- Stratégie IAM personnalisée - Pour les IPv6 clusters, créez et utilisez la politique IAM personnalisée suivante pour le VPC CNI. Cette politique est spécifique à Multi-NIC. Pour des informations plus générales sur l'utilisation du VPC CNI avec des IPv6 clusters, consultez. the section called "IPv6"

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSCNIPolicyIPv6MultiNIC",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:AssignIpv6Addresses",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeTags",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignIpv6Addresses",
                "ec2:AttachNetworkInterface",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonEKSCNIPolicyENITagIPv6MultiNIC",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        }
    ]
}
```

- IPv6mécanisme de transition non disponible - Si vous utilisez la fonctionnalité multi-NIC, le VPC CNI n'attribue pas IPv4 d'adresse aux pods d'un cluster. IPv6 Dans le cas contraire, le VPC CNI attribue une IPv4 adresse hôte locale à chaque pod afin qu'un pod puisse communiquer avec des ressources externes dans un IPv4 autre Amazon VPC ou sur Internet.

Utilisation

Une fois que la fonctionnalité multi-NIC est activée dans le CNI du VPC et `aws-node` que les pods ont redémarré, vous pouvez configurer chaque charge de travail pour qu'elle soit multihébergée. L'exemple suivant d'une configuration YAML avec les annotations requises :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders-deployment
  namespace: ecommerce
  labels:
    app: orders
spec:
  replicas: 3
  selector:
    matchLabels:
      app: orders
  template:
    metadata:
      annotations:
         k8s.amazonaws.com/nicConfig: multi-nic-attachment
      labels:
        app: orders
    spec:
...
```

Questions fréquentes (FAQ)

1. Qu'est-ce qu'une carte d'interface réseau (NIC) ?

Une carte d'interface réseau (NIC), également simplement appelée carte réseau, est un périphérique physique qui permet la connectivité réseau pour le matériel de cloud computing sous-jacent. Dans EC2 les serveurs modernes, cela fait référence à la carte réseau Nitro. Une interface réseau élastique (ENI) est une représentation virtuelle de cette carte réseau sous-jacente.

Certains types d' EC2 instances en ont plusieurs NICs pour améliorer les performances en termes de bande passante et de débit de paquets. Dans de tels cas, vous pouvez attribuer des cartes réseau secondaires ENIs aux cartes réseau supplémentaires. Par exemple, l'ENI #1 peut fonctionner comme interface pour la carte réseau attachée à l'index 0 de la carte réseau, tandis que l'ENI #2 peut fonctionner comme interface pour la carte réseau connectée à un index de carte réseau distinct.

2. Qu'est-ce qu'un module multilogement ?

Un pod multi-hébergement est un pod Kubernetes unique doté de plusieurs interfaces réseau (et par conséquent de plusieurs adresses IP). Chaque interface réseau du pod est associée à une interface réseau élastique (ENI), et il s' ENIs agit de représentations logiques distinctes NICs sur le nœud de travail sous-jacent. Avec plusieurs interfaces réseau, un module multiréseau dispose d'une capacité de transfert de données supplémentaire, ce qui augmente également son taux de transfert de données.

> **Important**
>
> Le VPC CNI peut uniquement configurer des pods multi-hôtes sur des types d'instances qui en ont plusieurs. NICs

3. Pourquoi utiliser cette fonctionnalité ?

Si vous devez adapter les performances réseau de vos charges de travail basées sur Kubernetes, vous pouvez utiliser la fonctionnalité multi-cartes réseau pour exécuter des pods multi-hôtes qui s'interfacent avec tous les composants sous-jacents NICs auxquels un périphérique ENA est connecté. L'utilisation de cartes réseau supplémentaires augmente la capacité de bande passante et les performances de débit de paquets de vos applications en répartissant le trafic des applications sur plusieurs connexions simultanées. Cela est particulièrement utile pour les cas d'utilisation de l'intelligence artificielle (IA), du Machine Learning (ML) et du calcul haute performance (HPC).

4. Comment utiliser cette fonctionnalité ?

1. Tout d'abord, vous devez vous assurer que votre cluster Kubernetes utilise VPC CNI version 1.20 ou ultérieure. Pour les étapes de mise à jour du VPC CNI en tant que module complémentaire EKS, consultez. the section called "Mise à jour (module complémentaire EKS)"
2. Ensuite, vous devez activer le support multi-NIC dans le VPC CNI à l'aide `ENABLE_MULTI_NIC` de la variable d'environnement.
3. Ensuite, vous devez vous assurer de créer et de joindre des nœuds dotés de plusieurs cartes réseau. Pour obtenir la liste des types d' EC2 instances dotés de plusieurs cartes réseau, consultez la section Cartes réseau du guide de EC2 l'utilisateur Amazon.
4. Enfin, vous configurez chaque charge de travail pour qu'elle utilise plusieurs interfaces réseau (pods multihébergés) ou qu'elle utilise une seule interface réseau.

5. Comment configurer mes charges de travail pour en utiliser plusieurs NICs sur un nœud de travail pris en charge ?

Pour utiliser des modules à plusieurs logements, vous devez ajouter l'annotation suivante :. `k8s.amazonaws.com/nicConfig: multi-nic-attachment` Cela associera un ENI de chaque carte réseau de l'instance sous-jacente au pod (un ou plusieurs mappages entre un pod et le NICs).

Si cette annotation est absente, le CNI VPC suppose que votre pod ne nécessite qu'une seule interface réseau et lui attribue une adresse IP provenant d'un ENI sur n'importe quelle carte réseau disponible.

6. Quels adaptateurs d'interface réseau sont compatibles avec cette fonctionnalité ?

Vous pouvez utiliser n'importe quel adaptateur d'interface réseau si au moins un ENA est connecté à la carte réseau sous-jacente pour le trafic IP. Pour plus d'informations sur l'ENA, consultez Elastic Network Adapter (ENA) dans le guide de EC2 l'utilisateur Amazon.

Configurations de périphériques réseau prises en charge :

- les interfaces ENA fournissent toutes les fonctionnalités de réseau et de routage IP traditionnelles requises pour prendre en charge le réseau IP d'un VPC. Pour plus d'informations, consultez Activer la mise en réseau améliorée avec ENA sur vos EC2 instances.
- Les interfaces EFA (EFA avec ENA) fournissent à la fois le dispositif ENA pour les réseaux IP et le dispositif EFA pour les communications à faible latence et haut débit.

> **Important**
>
> Si seul un adaptateur EFA est connecté à une carte réseau, le VPC CNI l'ignorera lors du provisionnement de la connectivité réseau pour un pod multiréseau. Toutefois, si vous combinez un adaptateur EFA uniquement avec un adaptateur ENA sur une carte réseau, le VPC CNI s'en chargera également sur cet appareil. ENIs Pour utiliser des interfaces EFA uniquement avec des clusters EKS, voir. the section called "Configurez des clusters de formation avec EFA"

7. Puis-je voir si un nœud de mon cluster est compatible avec l'ENA ?

Oui, vous pouvez utiliser la AWS CLI ou EC2 l'API pour récupérer des informations réseau concernant une EC2 instance de votre cluster. Cela indique si l'instance est compatible avec l'ENA ou non. Dans l'exemple suivant, remplacez `<your-instance-id>` par l'ID d' EC2 instance d'un nœud.

AWS Exemple de CLI :

```
aws ec2 describe-instances --instance-ids <your-instance-id> --query
 "Reservations[].Instances[].EnaSupport"
```

Exemple de sortie :

```
[ true ]
```

8. Puis-je voir les différentes adresses IP associées à un pod ?

Non, pas facilement. Cependant, vous pouvez utiliser `nsenter` depuis le nœud pour exécuter des outils réseau courants tels que `ip route show` et voir les adresses IP et interfaces supplémentaires.

9. Puis-je contrôler le nombre d'interfaces réseau pour mes pods ?

Non. Lorsque votre charge de travail est configurée pour en utiliser plusieurs NICs sur une instance prise en charge, un seul pod possède automatiquement une adresse IP provenant de chaque carte réseau de l'instance. Les pods à domicile unique seront également dotés d'une interface réseau connectée à une carte réseau de l'instance.

> **Important**
>
> Les cartes réseau auxquelles seul un périphérique EFA est connecté sont ignorées par le VPC CNI.

10. Puis-je configurer mes pods pour utiliser une carte réseau spécifique ?

Non, cela n'est pas pris en charge. Si un pod possède l'annotation appropriée, le VPC CNI le configure automatiquement pour utiliser chaque carte réseau dotée d'un adaptateur ENA sur le nœud de travail.

11. Cette fonctionnalité fonctionne-t-elle avec les autres fonctionnalités réseau VPC CNI ?

Oui, la fonctionnalité multi-NIC du VPC CNI fonctionne à la fois avec un réseau personnalisé et une découverte améliorée des sous-réseaux. Toutefois, les modules multihébergés n'utilisent pas les sous-réseaux ou groupes de sécurité personnalisés. Au lieu de cela, le VPC CNI attribue des adresses IP et des interfaces réseau aux pods multihébergés avec la même configuration de sous-réseau et de groupe de sécurité que le nœud. Pour plus d'informations sur la mise en réseau personnalisée, consultezthe section called "Mise en réseau personnalisée".

La fonctionnalité multi-NIC du VPC CNI ne fonctionne pas et ne peut pas être combinée avec les groupes de sécurité pour les pods.

12. Puis-je utiliser des politiques réseau avec cette fonctionnalité ?

Oui, vous pouvez utiliser les politiques réseau Kubernetes avec plusieurs cartes réseau. Les politiques réseau de Kubernetes limitent le trafic réseau à destination et en provenance de vos pods. Pour plus d'informations sur l'application de politiques réseau avec le VPC CNI, consultez. the section called "Politiques Kubernetes"

13. La prise en charge de plusieurs cartes d'interface réseau est-elle activée en mode automatique EKS ?

La multicarte réseau n'est pas prise en charge pour les clusters en mode automatique EKS.

Plug-ins CNI alternatifs pour les clusters Amazon EKS

Le plug-in Amazon VPC CNI pour Kubernetes est le seul plug-in CNI pris en charge par Amazon EKS avec des nœuds Amazon. EC2 Amazon EKS prend en charge les fonctionnalités de base de Cilium et Calico pour les nœuds hybrides Amazon EKS. Amazon EKS exécute Kubernetes en amont. Vous pouvez donc installer d'autres plug-ins CNI compatibles sur les EC2 nœuds Amazon de votre cluster. Si votre cluster contient des nœuds Fargate, le plug-in Amazon VPC CNI pour Kubernetes est déjà présent sur vos nœuds Fargate. C'est le seul plugin CNI que vous pouvez utiliser avec les nœuds Fargate. Une tentative d'installation d'un autre plugin CNI sur les nœuds Fargate a échoué.

Si vous envisagez d'utiliser un autre plug-in CNI sur les EC2 nœuds Amazon, nous vous recommandons d'obtenir un support commercial pour le plug-in ou de disposer de l'expertise interne nécessaire pour résoudre les problèmes et apporter des correctifs au projet de plug-in CNI.

Amazon EKS entretient des relations avec un réseau de partenaires qui offrent un support pour d'autres plugins CNI compatibles. Pour plus d'informations sur les versions, les qualifications et les tests effectués, consultez la documentation des partenaires suivante.

Partenaire	Produit (langue française non garantie)	Documentation
Tigera	Calico	Instructions d'installation
Isovalent	Cilium	Instructions d'installation
Juniper	Réseau Contrail natif dans le cloud () CN2	Instructions d'installation
VMware	Antrea	Instructions d'installation

Amazon EKS vise à vous offrir un large choix d'options pour couvrir tous les cas d'utilisation.

Autres plugins de politique réseau compatibles

Calico est une solution largement adoptée pour la mise en réseau et la sécurité des conteneurs. L'utilisation de Calico sur EKS permet d'appliquer une politique réseau totalement conforme à vos clusters EKS. En outre, vous pouvez choisir d'utiliser le réseau de Calico, qui conserve les adresses IP de votre VPC sous-jacent. Calico Cloud améliore les fonctionnalités de Calico Open Source en fournissant des fonctionnalités avancées de sécurité et d'observabilité.

Le flux de trafic à destination et en provenance des pods avec des groupes de sécurité associés n'est pas soumis à l'application de la politique réseau Calico et est limité à l'application des groupes de sécurité Amazon VPC uniquement.

Si vous appliquez la politique réseau Calico, nous vous recommandons de définir la variable d'environnement sur afin `ANNOTATE_POD_IP` d'`true`éviter un problème connu avec Kubernetes. Pour utiliser cette fonctionnalité, vous devez ajouter une `patch` autorisation pour les modules au `aws-node` ClusterRole. Notez que l'ajout d'autorisations de correctif `aws-node` DaemonSet augmente la portée de sécurité du plugin. Pour plus d'informations, consultez ANNOTATE_POD_IP dans le référentiel VPC CNI sur. GitHub

Considérations relatives au mode automatique d'Amazon EKS

Amazon EKS Auto Mode ne prend pas en charge les plug-ins CNI alternatifs ni les plug-ins de politique réseau. Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Connectez plusieurs interfaces réseau aux Pods avec Multus

Multus CNI est un plug-in d'interface réseau de conteneurs (CNI) pour Amazon EKS qui permet de connecter plusieurs interfaces réseau à un Pod. Pour plus d'informations, consultez la documentation Multus CNI sur GitHub.

Dans Amazon EKS, chaque pod possède une interface réseau attribuée par le plugin Amazon VPC CNI. Avec Multus, vous pouvez créer un Pod multi-hébergement doté de plusieurs interfaces. Ceci est accompli par Multus agissant comme un « méta-plugin » ; un plugin CNI qui peut appeler plusieurs autres plugins CNI. AWS le support de Multus est configuré avec le plugin Amazon VPC CNI comme plugin délégué par défaut.

- Amazon EKS ne créera ni ne publiera de plug-ins CNI pour I/O la virtualisation à racine unique (SR-IOV) et le kit de développement de plans de données (DPDK). Cependant, vous pouvez accélérer les paquets en vous connectant directement à Amazon EC2 Elastic Network Adapters (ENA) via un appareil hôte et des plugins gérés par Multus. `ipvlan`
- Amazon EKS prend en charge Multus, qui fournit un processus générique qui permet un chaînage simple de plugins CNI supplémentaires. Multus et le processus de chaînage sont pris en charge, mais AWS ne prendront pas en charge tous les plugins CNI compatibles qui peuvent être chaînés, ni les problèmes pouvant survenir dans les plugins CNI qui ne sont pas liés à la configuration de chaînage.
- Amazon EKS fournit une assistance et assure la gestion du cycle de vie du plug-in Multus, mais n'est pas responsable des adresses IP ou de la gestion supplémentaire associée aux interfaces réseau supplémentaires. L'adresse IP et la gestion de l'interface réseau par défaut utilisant le plugin CNI Amazon VPC restent inchangées.
- Seul le plugin CNI Amazon VPC est officiellement pris en charge en tant que plugin délégué par défaut. Vous devez modifier le manifeste d'installation de Multus publié pour reconfigurer le plugin délégué par défaut vers un autre CNI si vous choisissez de ne pas utiliser le plugin CNI Amazon VPC pour la mise en réseau principale.
- Multus n'est pris en charge que lorsque vous utilisez le CNI d'Amazon VPC comme CNI principal. Nous ne prenons pas en charge le CNI d'Amazon VPC lorsqu'il est utilisée pour des interfaces d'ordre supérieur, secondaires ou autres.
- Pour empêcher le plug-in Amazon VPC CNI d'essayer de gérer des interfaces réseau supplémentaires attribuées aux Pods, ajoutez la balise suivante à l'interface réseau :

  key

  : `node.k8s.amazonaws.com/no_manage`

valeur

: `true`

- Multus est compatible avec les politiques réseau, mais la politique doit être enrichie pour inclure les ports et les adresses IP qui peuvent faire partie d'interfaces réseau supplémentaires connectées aux Pods.

Pour une démonstration de la mise en œuvre, consultez le Guide d'installation de Multus sur GitHub.

Acheminez le trafic Internet avec le AWS Load Balancer Controller

Le AWS Load Balancer Controller gère les AWS Elastic Load Balancers pour un cluster Kubernetes. Vous pouvez utiliser le contrôleur pour exposer les applications de votre cluster à Internet. Le contrôleur fournit des équilibreurs de AWS charge qui pointent vers les ressources de service ou d'entrée du cluster. En d'autres termes, le contrôleur crée une adresse IP ou un nom DNS unique qui pointe vers plusieurs pods de votre cluster.

Le contrôleur surveille les ressources Kubernetes Ingress ou Service. En réponse, il crée les ressources AWS Elastic Load Balancing appropriées. Vous pouvez configurer le comportement spécifique des équilibreurs de charge en appliquant des annotations aux ressources Kubernetes. Par exemple, vous pouvez associer des groupes AWS de sécurité aux équilibreurs de charge à l'aide d'annotations.

Le contrôleur fournit les ressources suivantes :

Kubernetes **`Ingress`**

Le LBC crée un AWS Application Load Balancer (ALB) lorsque vous créez un Kubernetes. `Ingress` Passez en revue les annotations que vous pouvez appliquer à une ressource Ingress.

Service Kubernetes du type **`LoadBalancer`**

Le LBC crée un AWS Network Load Balancer (NLB) lorsque vous créez un service Kubernetes de ce type. `LoadBalancer` Passez en revue les annotations que vous pouvez appliquer à une ressource de service.

Dans le passé, l'équilibreur de charge réseau Kubernetes était utilisé pour les cibles d'instance, mais le LBC était utilisé pour les cibles IP. Avec la version AWS Load Balancer Controller `2.3.0` ou une version ultérieure, vous pouvez créer en NLBs utilisant l'un ou l'autre type de cible. Pour plus d'informations sur les types de cibles NLB, consultez Type de cible dans le guide de l'utilisateur de Network Load Balancer.

Le contrôleur est un projet open source géré sur GitHub.

Avant de déployer le contrôleur, nous vous recommandons de consulter les conditions préalables et les considérations des sections Router l'application et le trafic HTTP avec les équilibreurs de charge d'application et. the section called "Répartition de charge réseau" Dans ces rubriques, vous allez déployer un exemple d'application qui inclut un équilibreur de AWS charge.

Installez le contrôleur

Vous pouvez utiliser l'une des procédures suivantes pour installer le AWS Load Balancer Controller :

- Si vous utilisez Amazon EKS pour la première fois, nous vous recommandons d'utiliser Helm pour l'installation, car cela simplifie l'installation du AWS Load Balancer Controller. Pour de plus amples informations, veuillez consulter the section called "Installation avec Helm".
- Pour les configurations avancées, telles que les clusters avec un accès réseau restreint aux registres de conteneurs publics, utilisez les manifestes Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Installation avec des manifestes".

Migrer à partir de versions de contrôleurs obsolètes

- Si vous avez installé des versions obsolètes du AWS Load Balancer Controller, consultez. the section called "Migrer depuis une version obsolète"
- Les versions obsolètes ne peuvent pas être mises à niveau. Ils doivent être supprimés et une version actuelle du AWS Load Balancer Controller doit être installée.
- Les versions déconseillées incluent :

- AWS Contrôleur d'entrée ALB pour Kubernetes (« Ingress Controller »), prédécesseur du Load Balancer Controller. AWS
- N'importe quelle `0.1.x` version du AWS Load Balancer Controller

Fournisseur de cloud traditionnel

Kubernetes inclut un ancien fournisseur de cloud pour. AWS L'ancien fournisseur de cloud est capable de fournir des équilibreurs de AWS charge, comme le AWS Load Balancer Controller. L'ancien fournisseur de cloud crée des équilibreurs de charge classiques. Si vous n'installez pas le AWS Load Balancer Controller, Kubernetes utilisera par défaut l'ancien fournisseur de cloud. Vous devez installer le AWS Load Balancer Controller et éviter d'utiliser l'ancien fournisseur de cloud.

> **Important**
>
> Dans les versions 2.5 et ultérieures, le AWS Load Balancer Controller devient le contrôleur par défaut pour les ressources de service Kubernetes avec le Network Load Balancer (NLB) `type: LoadBalancer` et crée un AWS Network Load Balancer (NLB) pour chaque service. Pour ce faire, il génère un webhook de mutation pour les services, qui définit le champ `spec.loadBalancerClass` sur `service.k8s.aws/nlb` pour les nouveaux services d'équilibreur `type: LoadBalancer`. Vous pouvez désactiver cette fonctionnalité et revenir à l'utilisation de l'ancien fournisseur de cloud en tant que contrôleur par défaut, en définissant la valeur `enableServiceMutatorWebhook` des charts de Helm sur `false`. Le cluster ne fournira pas de nouveaux équilibreurs de charge classiques pour vos services à moins que vous ne désactiviez cette fonctionnalité. Les équilibreurs de charge Classic Load Balancer existants continueront de fonctionner.

Installez le contrôleur AWS Load Balancer avec Helm

> **Tip**
>
> Avec le mode automatique d'Amazon EKS, vous n'avez pas besoin d'installer ou de mettre à niveau des modules complémentaires réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Cette rubrique décrit comment installer le AWS Load Balancer Controller à l'aide de Helm, un gestionnaire de packages pour Kubernetes, et. `eksctl` Le contrôleur est installé avec les options par défaut. Pour plus d'informations sur le contrôleur, notamment sur sa configuration à l'aide d'annotations, consultez la documentation du contrôleur AWS Load Balancer sur. GitHub

Dans les étapes suivantes, remplacez *`example values`* par vos propres valeurs.

Prérequis

Avant de commencer ce didacticiel, vous devez suivre les étapes suivantes :

- Créez un cluster Amazon EKS. Pour en créer un, consultez Mise en route.
- Installez Helm sur votre machine locale.
- Assurez-vous que les versions minimales de votre plug-in Amazon VPC CNI pour Kubernetes et de vos modules complémentaires CoreDNS sont conformes aux versions minimales répertoriées dans les jetons de compte de service. `kube-proxy`
- Découvrez les concepts d' AWS Elastic Load Balancing. Pour plus d'informations, consultez le Guide de l'utilisateur Elastic Load Balancing.
- Découvrez le service Kubernetes et les ressources d'entrée.

Considérations

Avant de procéder aux étapes de configuration décrites sur cette page, tenez compte des points suivants :

- La politique et le rôle IAM (`AmazonEKSLoadBalancerControllerRole`) peuvent être réutilisés sur plusieurs clusters EKS du même AWS compte.
- Si vous installez le contrôleur sur le même cluster où le rôle (`AmazonEKSLoadBalancerControllerRole`) a été créé à l'origine, passez à l'étape 2 : Installer le Load Balancer Controller après avoir vérifié que le rôle existe.
- Si vous utilisez des rôles IAM pour les comptes de service (IRSA), l'IRSA doit être configuré pour chaque cluster, et l'ARN du fournisseur OpenID Connect (OIDC) figurant dans la politique de confiance du rôle est spécifique à chaque cluster EKS. En outre, si vous installez le contrôleur sur un nouveau cluster avec un contrôleur existant`AmazonEKSLoadBalancerControllerRole`, mettez à jour la politique de confiance du rôle pour inclure le fournisseur OIDC du nouveau cluster et créez un nouveau compte de service avec l'annotation de rôle appropriée. Pour déterminer si

vous avez déjà un fournisseur OIDC ou pour en créer un, consultezthe section called "Fournisseur IAM OIDC".

Étape 1 : créer un rôle IAM à l'aide de **eksctl**

Les étapes suivantes font référence à la version 2.13.3 de AWS Load Balancer Controller. Pour plus d'informations sur toutes les versions, consultez la page de mise à jour du AWS Load Balancer Controller sur. GitHub

1. Téléchargez une politique IAM pour le AWS Load Balancer Controller qui lui permet de passer des appels en votre AWS APIs nom.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.13.3/docs/install/iam_policy.json
```

   - Si vous êtes une AWS partition non standard, telle qu'une région gouvernementale ou chinoise, consultez les politiques GitHub et téléchargez la politique appropriée pour votre région.

2. Créez une politique IAM à l'aide de la politique téléchargée à l'étape précédente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Si vous consultez la politique dans le AWS Management Console, la console affiche des avertissements pour le service ELB, mais pas pour le service ELB v2. Cela est dû au fait que certaines des actions de la politique sont disponibles pour ELB v2, mais pas pour ELB. Vous pouvez ignorer les avertissements relatifs au service ELB.

3. Remplacez les valeurs du nom du cluster, du code de région et de l'ID de compte.

```
eksctl create iamserviceaccount \
    --cluster=<cluster-name> \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws: iam::<AWS_ACCOUNT_ID>:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
```

```
    --region <aws-region-code> \
    --approve
```

Étape 2 : Installation du AWS Load Balancer Controller

1. Ajoutez le référentiel de `eks-charts` diagrammes Helm. AWS maintient ce dépôt activé GitHub.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Mettez à jour votre référentiel local pour vous assurer que vous disposez des graphiques les plus récents.

```
helm repo update eks
```

3. Installez le AWS Load Balancer Controller.

   Si vous déployez le contrôleur sur des EC2 nœuds Amazon ayant un accès restreint au service de métadonnées d' EC2 instance Amazon (IMDS), ou si vous déployez sur des nœuds hybrides Fargate ou Amazon EKS, ajoutez les indicateurs suivants `helm` à la commande suivante :

   - `--set region=`*`region-code`*
   - `--set vpcId=`*`vpc-xxxxxxxx`*

     Remplacez *`my-cluster`* par le nom de votre cluster. Dans la commande suivante, `aws-load-balancer-controller` est le compte de service Kubernetes que vous avez créé à l'étape précédente.

     Pour plus d'informations sur la configuration du graphique de barre, consultez values.yaml on. GitHub

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --version 1.13.0
```

> **Important**
>
> Le graphique déployé ne reçoit pas automatiquement les mises à jour de sécurité. Vous devez effectuer manuellement une mise à niveau vers un graphique plus récent lorsqu'il sera disponible. Lors de la mise *install* à niveau, `upgrade` passez à la commande précédente.

La `helm install` commande installe automatiquement les définitions de ressources personnalisées (CRDs) pour le contrôleur. Ce n'est pas le cas de la `helm upgrade` commande. Si vous utilisez, `helm upgrade,` vous devez installer manuellement le CRDs. Exécutez la commande suivante pour installer CRDs :

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-
controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Étape 3 : vérifier que le contrôleur est installé

1. Vérifiez que le contrôleur est installé.

   ```
   kubectl get deployment -n kube-system aws-load-balancer-controller
   ```

   L'exemple qui suit illustre un résultat.

   ```
   NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
   aws-load-balancer-controller   2/2     2            2           84s
   ```

   Vous recevez la sortie précédente si vous avez déployé à l'aide de Helm. Si vous avez déployé à l'aide du manifeste Kubernetes, vous n'avez qu'un seul réplica.

2. Avant d'utiliser le contrôleur pour provisionner AWS des ressources, votre cluster doit répondre à des exigences spécifiques. Pour plus d'informations, consultez the section called "Répartition de la charge des applications" et the section called "Répartition de charge réseau".

Installer le AWS Load Balancer Controller avec des manifestes

> **Tip**
>
> Avec le mode automatique d'Amazon EKS, vous n'avez pas besoin d'installer ou de mettre à niveau des modules complémentaires réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Cette rubrique décrit comment installer le contrôleur en téléchargeant et en appliquant des manifestes Kubernetes. Vous pouvez afficher l'intégralité de la documentation pour le contrôleur sur GitHub.

Dans les étapes suivantes, remplacez *`example values`* par vos propres valeurs.

Prérequis

Avant de commencer ce didacticiel, vous devez suivre les étapes suivantes :

- Créez un cluster Amazon EKS. Pour en créer un, consultez Mise en route.
- Installez Helm sur votre machine locale.
- Assurez-vous que les versions minimales de votre plug-in Amazon VPC CNI pour Kubernetes et de vos modules complémentaires CoreDNS sont conformes aux versions minimales répertoriées dans les jetons de compte de service. `kube-proxy`
- Découvrez les concepts d' AWS Elastic Load Balancing. Pour plus d'informations, consultez le Guide de l'utilisateur Elastic Load Balancing.
- Découvrez le service Kubernetes et les ressources d'entrée.

Considérations

Avant de procéder aux étapes de configuration décrites sur cette page, tenez compte des points suivants :

- La politique et le rôle IAM (`AmazonEKSLoadBalancerControllerRole`) peuvent être réutilisés sur plusieurs clusters EKS du même AWS compte.

- Si vous installez le contrôleur sur le même cluster où le rôle (`AmazonEKSLoadBalancerControllerRole`) a été créé à l'origine, passez à l'étape 2 : Installer cert-manager après avoir vérifié que le rôle existe.
- Si vous utilisez des rôles IAM pour les comptes de service (IRSA), l'IRSA doit être configuré pour chaque cluster, et l'ARN du fournisseur OpenID Connect (OIDC) figurant dans la politique de confiance du rôle est spécifique à chaque cluster EKS. En outre, si vous installez le contrôleur sur un nouveau cluster avec un contrôleur existant`AmazonEKSLoadBalancerControllerRole`, mettez à jour la politique de confiance du rôle pour inclure le fournisseur OIDC du nouveau cluster et créez un nouveau compte de service avec l'annotation de rôle appropriée. Pour déterminer si vous avez déjà un fournisseur OIDC ou pour en créer un, consultezthe section called "Fournisseur IAM OIDC".

Étape 1 : Configuration de l'IAM

Les étapes suivantes font référence à la version 2.13.3 de AWS Load Balancer Controller. Pour plus d'informations sur toutes les versions, consultez la page de mise à jour du AWS Load Balancer Controller sur. GitHub

1. Téléchargez une politique IAM pour le AWS Load Balancer Controller qui lui permet de passer des appels en votre AWS APIs nom.

   Example

   AWS

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
   controller/v2.13.3/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
   controller/v2.13.3/docs/install/iam_policy_us-gov.json
   ```

   ```
   mv iam_policy_us-gov.json iam_policy.json
   ```

2. Créez une politique IAM à l'aide de la politique téléchargée à l'étape précédente.

   ```
   aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
   ```

```
    --policy-document file://iam_policy.json
```

> **Note**
>
> Si vous consultez la politique dans le AWS Management Console, la console affiche des avertissements pour le service ELB, mais pas pour le service ELB v2. Cela est dû au fait que certaines des actions de la politique sont disponibles pour ELB v2, mais pas pour ELB. Vous pouvez ignorer les avertissements relatifs au service ELB.

Example

eksctl

a. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par votre ID de compte, puis exécutez la commande.

```
eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

a. Récupérez l'ID du fournisseur OIDC de votre cluster et stockez-le dans une variable.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

b. Déterminez si un fournisseur IAM OIDC avec l'ID de votre cluster figure déjà dans votre compte. Vous devez configurer OIDC à la fois pour le cluster et pour IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Si une sortie est renvoyée, cela signifie que vous disposez déjà d'un fournisseur IAM OIDC pour votre cluster. Si aucune sortie n'est renvoyée, vous devez créer un fournisseur OIDC IAM pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Fournisseur IAM OIDC".

c. Copiez les contenus suivants sur votre appareil. Remplacez *111122223333* par votre ID de compte. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster. Remplacez *EXAMPLED539D4633E53DE1B71EXAMPLE* par la sortie renvoyée dans l'étape précédente.

```
cat >load-balancer-role-trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws: iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

d. Créez le rôle IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. Attachez la politique IAM gérée par Amazon EKS au rôle IAM. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

f. Copiez les contenus suivants sur votre appareil. Remplacez *111122223333* par votre ID de compte. Après avoir remplacé le texte, exécutez la commande modifiée pour créer le fichier `aws-load-balancer-controller-service-account.yaml`.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKSLoadBalancerControllerRole
EOF
```

g. Créez le compte de service Kubernetes sur votre cluster. Le compte de service Kubernetes nommé `aws-load-balancer-controller` est annoté avec le nom du rôle IAM que vous avez créé. *AmazonEKSLoadBalancerControllerRole*

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Étape 2 : Installation **`cert-manager`**

Installez `cert-manager` en utilisant l'une des méthodes suivantes pour injecter la configuration du certificat dans les webhooks. Pour plus d'informations, consultez Getting Started dans la documentation de cert-manager.

Nous vous recommandons d'utiliser le registre des `quay.io` conteneurs pour l'installation`cert-manager`. Si vos nœuds n'ont pas accès au registre des `quay.io` conteneurs, installez-le à `cert-manager` l'aide d'Amazon ECR (voir ci-dessous).

Example

Quay.io

a. Si vos nœuds ont accès au registre de conteneurs `quay.io`, installez `cert-manager` pour injecter la configuration du certificat dans les webhooks.

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

Amazon ECR

a. Installez `cert-manager` en utilisant l'une des méthodes suivantes pour injecter la configuration du certificat dans les webhooks. Pour plus d'informations, consultez Getting Started dans la documentation de cert-manager.

b. Téléchargez le manifeste.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

c. Extrayez les images suivantes et transférez-les vers un référentiel auquel vos nœuds ont accès. Pour plus d'informations sur l'extraction, le balisage et le transfert d'images vers votre propre référentiel, consultez the section called "Copier une image dans un référentiel".

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

d. Remplacez `quay.io` dans le manifeste des trois images par votre propre nom de registre. La commande suivante suppose que le nom de votre dépôt privé est le même que celui du dépôt source. Remplacez *`111122223333.dkr.ecr.region-code.amazonaws.com`* par votre registre privé.

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-manager.yaml
```

e. Appliquez le manifeste.

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

Étape 3 : Installation du AWS Load Balancer Controller

1. Téléchargez la spécification du contrôleur. Pour plus d'informations sur le contrôleur, consultez la documentation sur GitHub.

```
curl -Lo v2_13_3_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.13.3/v2_13_3_full.yaml
```

2. Effectuez les modifications suivantes dans le fichier.

   a. Si vous avez téléchargé le fichier `v2_13_3_full.yaml`, exécutez la commande suivante pour supprimer la section `ServiceAccount` du manifeste. Si vous ne supprimez pas cette section, l'annotation requise que vous avez apportée au compte de service lors d'une étape précédente est remplacée. La suppression de cette section préserve également le compte de service que vous avez créé à une étape précédente, si vous supprimez le contrôleur.

```
sed -i.bak -e '690,698d' ./v2_13_3_full.yaml
```

   Si vous avez téléchargé une version différente du fichier, ouvrez le fichier dans un éditeur et supprimez les lignes suivantes.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

b. Remplacez `your-cluster-name` dans la section `Deployment spec` du fichier par le nom de votre cluster en remplaçant *`my-cluster`* par le nom de votre cluster.

```
sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_13_3_full.yaml
```

c. Si vos nœuds n'ont pas accès aux référentiels d'images Amazon ECR Amazon EKS, vous devez extraire l'image suivante et la transférer vers un référentiel auquel vos nœuds ont accès. Pour plus d'informations sur l'extraction, l'identification et le transfert d'une image vers votre propre référentiel, consultez the section called "Copier une image dans un référentiel".

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.13.3
```

Ajoutez le nom de votre registre au manifeste. La commande suivante suppose que le nom de votre dépôt privé est le même que celui du dépôt source et ajoute le nom de votre registre privé au fichier. Remplacez *`111122223333.dkr.ecr.region-code.amazonaws.com`* par votre registre. Cette ligne suppose que vous avez nommé votre référentiel privé de la même manière que le référentiel source. Dans le cas contraire, remplacez le texte `eks/aws-load-balancer-controller` après votre nom de registre privé par le nom de votre référentiel.

```
sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-controller|
111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-balancer-
controller|' ./v2_13_3_full.yaml
```

d. (Requis uniquement pour Fargate ou Restricted IMDS)

Si vous déployez le contrôleur sur des EC2 nœuds Amazon qui ont un accès restreint au service de métadonnées d' EC2 instance Amazon (IMDS), ou si vous déployez sur des nœuds hybrides Fargate ou Amazon EKS, ajoutez le point ci-dessous. `following parameters -args:`

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code
```

```
[...]
```

3. Appliquez le fichier.

```
kubectl apply -f v2_13_3_full.yaml
```

4. Téléchargez l'attaque de l'homme du milieu (HDM) `IngressClass` et `IngressClassParams` à votre cluster.

```
curl -Lo v2_13_3_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.13.3/v2_13_3_ingclass.yaml
```

5. Appliquez le manifeste à votre cluster.

```
kubectl apply -f v2_13_3_ingclass.yaml
```

Étape 4 : vérifier que le contrôleur est installé

1. Vérifiez que le contrôleur est installé.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

L'exemple qui suit illustre un résultat.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Vous recevez la sortie précédente si vous avez déployé à l'aide de Helm. Si vous avez déployé à l'aide du manifeste Kubernetes, vous n'avez qu'un seul réplica.

2. Avant d'utiliser le contrôleur pour provisionner AWS des ressources, votre cluster doit répondre à des exigences spécifiques. Pour plus d'informations, consultez the section called "Répartition de la charge des applications" et the section called "Répartition de charge réseau".

Migrer des applications depuis le contrôleur d'entrée ALB obsolète

Cette rubrique décrit comment effectuer une migration à partir de versions de contrôleurs obsolètes. Plus précisément, il décrit comment supprimer les versions obsolètes du AWS Load Balancer Controller.

- Les versions obsolètes ne peuvent pas être mises à niveau. Vous devez d'abord les supprimer, puis installer une version actuelle.
- Les versions déconseillées incluent :
  - AWS Contrôleur d'entrée ALB pour Kubernetes (« Ingress Controller »), prédécesseur du Load Balancer Controller. AWS
  - N'importe quelle `0.1.x` version du AWS Load Balancer Controller

Supprimer la version obsolète du contrôleur

> **Note**
>
> Vous avez peut-être installé la version obsolète à l'aide de Helm ou manuellement à l'aide de manifestes Kubernetes. Terminez la procédure à l'aide de l'outil avec lequel vous l'avez installé à l'origine.

1. Si vous avez installé les Charts de Helm `incubator/aws-alb-ingress-controller`, désinstallez-les.

```
helm delete aws-alb-ingress-controller -n kube-system
```

2. Si la version `0.1.x` du graphique `eks-charts/aws-load-balancer-controller` est installée, désinstallez-la. La mise à niveau de la version `0.1.x` vers `1.0.0` ne fonctionne pas en raison d'une incompatibilité avec la version de l'API Webhook.

```
helm delete aws-load-balancer-controller -n kube-system
```

3. Vérifiez si le contrôleur est actuellement installé.

```
kubectl get deployment -n kube-system alb-ingress-controller
```

   Il s'agit de la sortie si le contrôleur n'est pas installé.

```
Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
```

   Il s'agit de la sortie si le contrôleur est installé.

```
NAME                    READY UP-TO-DATE AVAILABLE AGE
alb-ingress-controller 1/1   1          1         122d
```

4. Saisissez les commandes suivantes pour supprimer le contrôleur.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/rbac-role.yaml
```

Migrer vers AWS Load Balancer Controller

Pour migrer de l'ALB Ingress Controller pour Kubernetes vers le Load Balancer Controller, vous AWS devez :

1. Retirez le contrôleur d'entrée ALB (voir ci-dessus).
2. Installez le AWS Load Balancer Controller.
3. Ajoutez une politique supplémentaire au rôle IAM utilisé par le AWS Load Balancer Controller. Cette politique permet au LBC de gérer les ressources créées par le contrôleur d'entrée ALB pour Kubernetes.
4. Téléchargez la politique IAM. Cette politique permet au AWS Load Balancer Controller de gérer les ressources créées par le contrôleur ALB Ingress pour Kubernetes. Vous pouvez également afficher la politique.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.13.3/docs/install/iam_policy_v1_to_v2_additional.json
```

5. Si votre cluster se trouve dans les AWS régions AWS GovCloud (USA Est) ou AWS GovCloud (USA Ouest), remplacez-le par `arn:aws:` . `arn:aws-us-gov:`

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. Créez la politique IAM et notez l'ARN renvoyé.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. Attachez la politique IAM au rôle IAM utilisé par le AWS Load Balancer Controller. Remplacez *your-role-name* par le nom du rôle, tel que`AmazonEKSLoadBalancerControllerRole`.

   Si vous avez créé le rôle en utilisant`eksctl`, pour trouver le nom du rôle créé, ouvrez la AWS CloudFormation console et sélectionnez la pile eksctl- ***my-cluster*** - addon-iamserviceaccount-kube-system - aws-load-balancer-controller. Sélectionnez l'onglet Resources (Ressources). Le nom du rôle se trouve dans la colonne ID Physique.

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

Gérer CoreDNS pour DNS dans les clusters Amazon EKS

> **Tip**
>
> Avec le mode automatique d'Amazon EKS, vous n'avez pas besoin d'installer ou de mettre à niveau des modules complémentaires réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

CoreDNS est un serveur DNS flexible et extensible qui peut servir de DNS de cluster Kubernetes. Lorsque vous lancez un cluster Amazon EKS avec au moins un nœud, deux réplicas de l'image CoreDNS sont déployés par défaut, quel que soit le nombre de nœuds déployés dans votre cluster. Les pods CoreDNS fournissent une résolution de noms pour tous les pods du cluster. Les pods CoreDNS peuvent être déployés sur les nœuds Fargate si votre cluster inclut un profil Fargate avec un espace de noms qui correspond à l'espace de noms du CoreDNS. `deployment` Pour plus d'informations sur CoreDNS, consultez Utilisation de CoreDNS pour la découverte des services dans la documentation Kubernetes.

Versions de CoreDNS

Le tableau suivant répertorie la dernière version du type de module complémentaire Amazon EKS pour chaque version de Kubernetes.

Version de Kubernetes	Version CoreDNS
1,33	v1.12.2-eksbuild.4
1,32	v1.11.4-eksbuild.14
1,31	v1.11.4-eksbuild.14
1,30	v1.11.4-eksbuild.14
1,29	v1.11.4-eksbuild.14
1,28	v1.10.1-eksbuild.35
1,27	v1.10.1-eksbuild.35
1,26	v1.9.3-eksbuild.39

> **Important**
>
> Si vous gérez vous-même ce module complémentaire, les versions du tableau ne sont peut-être pas les mêmes que les versions autogérées disponibles. Pour plus d'informations sur la mise à jour du type autogéré de ce module complémentaire, consultez la rubrique the section called "Mise à jour (autogérée)".

Considérations importantes relatives à la mise à niveau de CoreDNS

- Les mises à jour CoreDNS utilisent PodDisruptionBudget un pour maintenir la disponibilité du service DNS pendant le processus de mise à jour.
- Pour améliorer la stabilité et la disponibilité du déploiement de CoreDNS, les `v1.9.3-eksbuild.6` versions `v1.10.1-eksbuild.3` et ultérieures sont déployées avec un. `PodDisruptionBudget` Si vous avez déployé une version existante`PodDisruptionBudget`, votre mise à niveau vers ces versions risque d'échouer. Si la mise à niveau échoue, l'exécution de l'une des tâches suivantes devrait résoudre le problème :
  - Lors de la mise à niveau du module complémentaire Amazon EKS, choisissez de remplacer les paramètres existants comme option de résolution des conflits. Si vous avez défini d'autres

paramètres personnalisés pour le déploiement, assurez-vous de les sauvegarder avant de procéder à la mise à niveau afin de pouvoir réappliquer vos autres paramètres personnalisés après la mise à niveau.

- Supprimez votre version existante `PodDisruptionBudget` et réessayez d'effectuer la mise à niveau.

- Dans les versions complémentaires d'EKS `v1.9.3-eksbuild.3` et versions ultérieures `v1.10.1-eksbuild.6` et ultérieures, le déploiement CoreDNS définit le pour utiliser `readinessProbe` le point de terminaison. `/ready` Ce point de terminaison est activé dans le fichier `Corefile` de configuration de CoreDNS.

  Si vous utilisez un plugin personnalisé`Corefile`, vous devez ajouter le `ready` plugin à la configuration, afin que le `/ready` point de terminaison soit actif dans CoreDNS pour que la sonde puisse l'utiliser.

- Dans les versions `v1.9.3-eksbuild.7` et ultérieures, et `v1.10.1-eksbuild.4` et ultérieures des modules complémentaires EKS, vous pouvez modifier le `PodDisruptionBudget`. Vous pouvez modifier le module complémentaire et modifier ces paramètres dans les paramètres de configuration facultatifs à l'aide des champs de l'exemple suivant. Cet exemple montre la valeur `PodDisruptionBudget` par défaut.

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

  Vous pouvez définir `maxUnavailable` ou`minAvailable`, mais vous ne pouvez pas définir les deux en une seule fois`PodDisruptionBudget`. Pour plus d'informations`PodDisruptionBudgets`, consultez la section Spécifier un PodDisruptionBudget dans la documentation de Kubernetes.

  Notez que si vous définissez sur `enabledfalse`, le `PodDisruptionBudget` n'est pas supprimé. Après avoir défini ce champ sur `false`, vous devez supprimer l'objet `PodDisruptionBudget`. De même, si vous modifiez le module complémentaire pour utiliser une ancienne version du module complémentaire (rétrogradez le module complémentaire) après la mise à niveau vers une version avec un`PodDisruptionBudget`, celui-ci `PodDisruptionBudget` n'est pas supprimé. Pour supprimer le `PodDisruptionBudget`, exécutez la commande suivante :

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- Dans les versions complémentaires d'EKS `v1.10.1-eksbuild.5` et versions ultérieures, modifiez la tolérance par défaut de `node-role.kubernetes.io/master:NoSchedule` à pour vous conformer `node-role.kubernetes.io/control-plane:NoSchedule` à KEP 2067. Pour plus d'informations sur KEP 2067, voir KEP-2067 : Renommer l'étiquette « master » de kubeadm et altérer les propositions d'amélioration de Kubernetes () sur. KEPs GitHub

  Dans les versions complémentaires d'EKS `v1.8.7-eksbuild.8` et les versions ultérieures, les deux tolérances sont définies pour être compatibles avec toutes les versions de Kubernetes. `v1.9.3-eksbuild.9`

- Dans les versions complémentaires d'EKS `v1.9.3-eksbuild.11` `v1.10.1-eksbuild.7` et versions ultérieures, le déploiement CoreDNS définit une valeur par défaut pour. `topologySpreadConstraints` La valeur par défaut garantit que les pods CoreDNS sont répartis dans les zones de disponibilité s'il existe des nœuds disponibles dans plusieurs zones de disponibilité. Vous pouvez définir une valeur personnalisée qui sera utilisée à la place de la valeur par défaut. La valeur par défaut est la suivante :

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Considérations relatives à la mise à niveau de **`v1.11`** CoreDNS

- Dans les versions complémentaires d'EKS `v1.11.1-eksbuild.4` et versions ultérieures, l'image du conteneur est basée sur une image de base minimale gérée par Amazon EKS Distro, qui contient un minimum de packages et ne possède pas de shell. Pour plus d'informations, consultez Amazon EKS Distro. L'utilisation et le dépannage de l'image CoreDNS restent les mêmes.

Création du module complémentaire CoreDNS Amazon EKS

Créez le module complémentaire CoreDNS Amazon EKS. Vous devez disposer d'un cluster avant de créer le module complémentaire. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster ".

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster.

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
d : -f 3
```

   L'exemple qui suit illustre un résultat.

```
v1.10.1-eksbuild.13
```

2. Déterminez le type de module complémentaire installé sur votre cluster. Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

   Si un numéro de version est renvoyé, le type Amazon EKS du module complémentaire est installé sur votre cluster et vous n'avez pas besoin de suivre les étapes restantes de cette procédure. Si une erreur est renvoyée, le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Suivez les étapes restantes de cette procédure pour l'installer.

3. Enregistrez la configuration du module complémentaire actuellement installé.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. Créez le module complémentaire à l'aide de la AWS CLI. Si vous souhaitez utiliser le AWS Management Console ou `eksctl` pour créer le module complémentaire, consultez the section called "Création d'un module complémentaire" et spécifiez `coredns` le nom du module complémentaire. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.
   - Remplacez *my-cluster* par le nom de votre cluster.
   - Remplacez *v1.11.3-eksbuild.1* par la dernière version répertoriée dans le tableau des dernières versions pour la version de votre cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1
```

Si vous avez appliqué des paramètres personnalisés à votre module complémentaire actuel qui entrent en conflit avec les paramètres par défaut du module complémentaire Amazon EKS, la création risque d'échouer. Si la création échoue, vous recevez un message d'erreur qui peut vous aider à résoudre le problème. Vous pouvez également ajouter `--resolve-conflicts OVERWRITE` à la commande précédente. Cela permet au module complémentaire de remplacer les paramètres personnalisés existants. Une fois que vous avez créé le module complémentaire, vous pouvez le mettre à jour avec vos paramètres personnalisés.

5. Vérifiez que la dernière version du module complémentaire pour la version Kubernetes de votre cluster a été ajoutée à votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

La création du module complémentaire peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
v1.11.3-eksbuild.1
```

6. Si vous avez personnalisé les paramètres du module complémentaire, avant de créer le module complémentaire Amazon EKS, utilisez la configuration que vous avez enregistrée lors d'une étape précédente pour mettre à jour le module complémentaire Amazon EKS avec vos paramètres personnalisés. Pour obtenir des instructions sur la mise à jour du module complémentaire, consultezthe section called “Mise à jour (module complémentaire EKS)”.

Mettre à jour le module complémentaire CoreDNS Amazon EKS

Mettez à jour le type de module complémentaire Amazon EKS. Si vous n'avez pas ajouté le module complémentaire Amazon EKS à votre cluster, ajoutez-le ou consultezthe section called “Mise à jour (autogérée)”.

Avant de commencer, passez en revue les considérations relatives à la mise à niveau. Pour de plus amples informations, veuillez consulter the section called “Considérations importantes relatives à la mise à niveau de CoreDNS”.

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

L'exemple qui suit illustre un résultat.

```
v1.10.1-eksbuild.13
```

Si la version renvoyée est identique à la version de Kubernetes de votre cluster indiquée dans le tableau des dernières versions, cela signifie que la dernière version est déjà installée sur votre cluster et vous n'avez pas besoin de suivre le reste de cette procédure. Si vous recevez une erreur, au lieu d'un numéro de version dans votre sortie, cela signifie que le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Vous devez créer le module complémentaire avant de pouvoir le mettre à jour à l'aide de cette procédure.

2. Enregistrez la configuration du module complémentaire actuellement installé.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. Mettez à jour votre module complémentaire à l'aide de la AWS CLI. Si vous souhaitez utiliser AWS Management Console ou mettre `eksctl` à jour le module complémentaire, consultezthe section called "Mettre à jour un module complémentaire". Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.

   - Remplacez *my-cluster* par le nom de votre cluster.
   - Remplacez *v1.11.3-eksbuild.1* par la dernière version répertoriée dans le tableau des dernières versions pour la version de votre cluster.
   - L'`--resolve-conflicts`*PRESERVE* option préserve les valeurs de configuration existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres complémentaires et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `OVERWRITE`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour des paramètres, il est possible qu'elles

soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur parnone, Amazon EKS ne modifie pas la valeur des paramètres, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.

- Si vous ne mettez pas à jour un paramètre de configuration, `--configuration-values '{"replicaCount":3}'` supprimez-le de la commande. Si vous mettez à jour un paramètre de configuration, `"replicaCount":3` remplacez-le par le paramètre que vous souhaitez définir. Dans cet exemple, le nombre de répliques de CoreDNS est défini sur. 3 La valeur que vous spécifiez doit être valide pour le schéma de configuration. Si vous ne connaissez pas le schéma de configuration`aws eks describe-addon-configuration --addon-name coredns --addon-version v1.11.3-eksbuild.1` , exécutez-le en le `v1.11.3-eksbuild.1` remplaçant par le numéro de version du module complémentaire dont vous souhaitez consulter la configuration. Le schéma est renvoyé dans la sortie. Si vous avez une configuration personnalisée existante, que vous souhaitez tout supprimer et rétablir les valeurs de tous les paramètres sur les valeurs par défaut d'Amazon EKS, supprimez-la `"replicaCount":3` de la commande afin d'avoir le champ vide`{}`. Pour plus d'informations sur les paramètres CoreDNS, consultez la section Personnalisation du service DNS dans la documentation de Kubernetes.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

  La mise à jour peut prendre plusieurs secondes.

4. Assurez-vous que la version du module complémentaire a été mise à jour. Remplacez `my-cluster` par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

La mise à jour peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
```

```
        "addonVersion": "v1.11.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws: eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

Mettre à jour le module complémentaire autogéré CoreDNS Amazon EKS

> **Important**
>
> Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si vous ne connaissez pas la différence entre les types, consultezthe section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Création d'un module complémentaire". Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

Avant de commencer, passez en revue les considérations relatives à la mise à niveau. Pour de plus amples informations, veuillez consulter the section called "Considérations importantes relatives à la mise à niveau de CoreDNS".

1. Vérifiez que le type autogéré du module complémentaire est installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Si un message d'erreur est renvoyé, le type autogéré du module complémentaire est installé sur votre cluster. Suivez les étapes restantes de cette procédure. Si un numéro de version est

renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Pour mettre à jour le type de module complémentaire Amazon EKS, utilisez la procédure décrite dans Mettre à jour le module complémentaire Amazon EKS CoreDNS, plutôt que d'utiliser cette procédure. Si vous ne connaissez pas les différences entre les types de modules complémentaires, consultezthe section called "Modules complémentaires Amazon EKS".

2. Découvrez quelle version de l'image de conteneur est actuellement installée sur votre cluster.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.8.7-eksbuild.2
```

3. Si votre version actuelle de CoreDNS `v1.5.0` est ou ultérieure, mais antérieure à la version répertoriée dans le tableau des versions de CoreDNS, ignorez cette étape. Si votre version actuelle est antérieure à`1.5.0`, vous devez modifier le module `ConfigMap` pour que CoreDNS utilise le module complémentaire de transfert plutôt que le module complémentaire de proxy.

   a. Ouvrez le `ConfigMap` avec la commande suivante.

   ```
   kubectl edit configmap coredns -n kube-system
   ```

   b. Remplacez `proxy` dans la ligne suivante par `forward`. Enregistrez le fichier et quittez l'éditeur.

   ```
   proxy . /etc/resolv.conf
   ```

4. Si vous avez initialement déployé votre cluster sur Kubernetes `1.17` ou une version antérieure, vous devrez peut-être supprimer une ligne interrompue de votre manifeste CoreDNS.

   > **Important**
   >
   > Vous devez effectuer cette étape avant de passer à la `1.7.0` version CoreDNS, mais il est recommandé de procéder à cette étape même si vous effectuez une mise à jour vers une version antérieure.

   a. Vérifiez si votre manifeste CoreDNS possède la ligne.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

Si aucune sortie n'est renvoyée, votre manifeste ne contient pas la ligne et vous pouvez passer à l'étape suivante pour mettre à jour CoreDNS. Si la sortie est renvoyée, vous devez supprimer la ligne.

b. Modifiez la `ConfigMap` en utilisant la commande suivante, en supprimant la ligne dans le fichier contenant le mot `upstream`. Ne modifiez rien d'autre dans le fichier. Une fois la ligne supprimée, enregistrez les modifications.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Récupérez la version actuelle de votre image CoreDNS :

```
kubectl describe deployment coredns -n kube-system | grep Image
```

L'exemple qui suit illustre un résultat.

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Si vous effectuez une mise à jour vers `1.8.3` CoreDNS ou une version ultérieure, vous devez ajouter l'autorisation à `endpointslices` Kubernetes. `system:coredns` `clusterrole`

```
kubectl edit clusterrole system:coredns -n kube-system
```

Ajoutez les lignes suivantes sous les lignes d'autorisation existantes dans la section `rules` du fichier.

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. Mettez à jour le module complémentaire CoreDNS en *602401143452* remplaçant *region-code* et par les valeurs de la sortie renvoyée à une étape précédente. *v1.11.3-eksbuild.1*Remplacez-le par la version de CoreDNS répertoriée dans le tableau des dernières versions de votre version de Kubernetes.

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.3-
eksbuild.1
```

L'exemple qui suit illustre un résultat.

```
deployment.apps/coredns image updated
```

8. Vérifiez à nouveau la version d'image de conteneur pour confirmer qu'elle a été mise à jour vers la version que vous avez spécifiée à l'étape précédente.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.11.3-eksbuild.1
```

Adaptez les pods CoreDNS à un trafic DNS élevé

Lorsque vous lancez un cluster Amazon EKS avec au moins un nœud, un déploiement de deux répliques de l'image CoreDNS est déployé par défaut, quel que soit le nombre de nœuds déployés dans votre cluster. Les pods CoreDNS fournissent une résolution de noms pour tous les pods du cluster. Les applications utilisent la résolution de noms pour se connecter aux pods et aux services du cluster ainsi qu'aux services extérieurs au cluster. À mesure que le nombre de demandes de résolution de noms (requêtes) provenant des pods augmente, les pods CoreDNS peuvent être débordés et ralentir, et rejeter les demandes qu'ils ne peuvent pas traiter.

Pour gérer la charge accrue sur les pods CoreDNS, envisagez un système de mise à l'échelle automatique pour CoreDNS. Amazon EKS peut gérer le dimensionnement automatique du déploiement de CoreDNS dans la version complémentaire EKS de CoreDNS. Cet autoscaler CoreDNS surveille en permanence l'état du cluster, y compris le nombre de nœuds et de cœurs de processeur. Sur la base de ces informations, le contrôleur adaptera dynamiquement le nombre

de répliques du déploiement CoreDNS dans un cluster EKS. Cette fonctionnalité fonctionne pour CoreDNS et `v1.9` versions ultérieures. Pour plus d'informations sur les versions compatibles avec CoreDNS Autoscaling, consultez la section suivante.

Le système gère automatiquement les répliques CoreDNS à l'aide d'une formule dynamique basée à la fois sur le nombre de nœuds et de cœurs de processeur dans le cluster, calculé comme le maximum de numberOfNodes (divisé par 16) et (CPUCores NumberOf divisé par 256). Il évalue la demande sur des périodes de pointe de 10 minutes, augmente immédiatement lorsque cela est nécessaire pour gérer l'augmentation de la charge de requêtes DNS, tout en diminuant progressivement en réduisant le nombre de répliques de 33 % toutes les 3 minutes afin de maintenir la stabilité du système et d'éviter les perturbations.

Nous vous recommandons d'utiliser cette fonctionnalité conjointement avec d'autres bonnes pratiques d'EKS Cluster Autoscaling afin d'améliorer la disponibilité globale des applications et l'évolutivité du cluster.

Prérequis

Pour qu'Amazon EKS puisse adapter votre déploiement CoreDNS, trois conditions préalables sont requises :

- Vous devez utiliser la version complémentaire EKS de CoreDNS.
- Votre cluster doit exécuter au moins les versions de cluster et les versions de plate-forme minimales.
- Votre cluster doit exécuter au moins la version minimale du module complémentaire EKS de CoreDNS.

Version minimale du cluster

Le dimensionnement automatique de CoreDNS est effectué par un nouveau composant du plan de contrôle du cluster, géré par Amazon EKS. Pour cette raison, vous devez mettre à niveau votre cluster vers une version d'EKS prenant en charge la version minimale de plate-forme contenant le nouveau composant.

Un nouveau cluster Amazon EKS. Pour en déployer un, consultez Mise en route. Le cluster doit exécuter l'une des versions de Kubernetes et des versions de plate-forme répertoriées dans le tableau suivant ou une version ultérieure. Notez que toutes les versions de Kubernetes et de plate-forme ultérieures à celles répertoriées sont également prises en charge. Vous pouvez vérifier votre

version actuelle de Kubernetes *my-cluster* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Version de Kubernetes	Version de la plateforme
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18

> **Note**
>
> Toutes les versions de plate-forme des versions ultérieures de Kubernetes sont également prises en charge, par exemple les versions ultérieures de Kubernetes. 1.30 eks.1

Version minimale du module complémentaire EKS

Version de Kubernetes	1,29	1,28	1,27	1,26
	v1.11.1-eksbuild.9	v1.10.1-eksbuild.11	v1.10.1-eksbuild.11	v1.9.3-eksbuild.15

Configuration de la mise à l'échelle automatique de CoreDNS dans le AWS Management Console

1. Assurez-vous que votre cluster est égal ou supérieur à la version minimale du cluster.

   Amazon EKS met automatiquement à niveau les clusters entre les versions de plate-forme d'une même version de Kubernetes, et vous ne pouvez pas démarrer ce processus vous-même. Au lieu de cela, vous pouvez mettre à niveau votre cluster vers la prochaine version de Kubernetes, et le cluster sera mis à niveau vers cette version K8s et la dernière version de la plateforme.

Les nouvelles versions de Kubernetes introduisent parfois des modifications importantes. Par conséquent, nous vous recommandons de tester le comportement de vos applications en utilisant un cluster distinct de la nouvelle version de Kubernetes avant de mettre à jour vos clusters de production.

Pour mettre à niveau un cluster vers une nouvelle version de Kubernetes, suivez la procédure décrite dans Mettre à jour un cluster existant vers une nouvelle version de Kubernetes.

2. Assurez-vous que vous disposez du module complémentaire EKS pour CoreDNS, et non du déploiement CoreDNS autogéré.

   Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Pour savoir quel type de module complémentaire est installé sur votre cluster, vous pouvez exécuter la commande suivante. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Si un numéro de version est renvoyé, le type Amazon EKS du module complémentaire est installé sur votre cluster et vous pouvez passer à l'étape suivante. Si une erreur est renvoyée, le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Effectuez les étapes restantes de la procédure Créez le module complémentaire CoreDNS Amazon EKS pour remplacer la version autogérée par le module complémentaire Amazon EKS.

3. Assurez-vous que la version de votre module complémentaire EKS pour CoreDNS est identique ou supérieure à la version minimale du module complémentaire EKS.

   Déterminez la version du module complémentaire actuellement installée sur votre cluster. Vous pouvez enregistrer AWS Management Console ou exécuter la commande suivante :

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.10.1-eksbuild.13
   ```

Comparez cette version avec la version minimale du module complémentaire EKS dans la section précédente. Si nécessaire, mettez à niveau le module complémentaire EKS vers une version supérieure en suivant la procédure Mettre à jour le module complémentaire CoreDNS Amazon EKS.

4. Ajoutez la configuration de mise à l'échelle automatique aux paramètres de configuration facultatifs du module complémentaire EKS.
   a. Ouvrez la console Amazon EKS.
   b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire.
   c. Choisissez l'onglet Modules complémentaires.
   d. Cochez la case en haut à droite de la zone du module complémentaire CoreDNS, puis choisissez Modifier.
   e. Sur la page Configurer CoreDNS :
      i. Sélectionnez la version que vous souhaitez utiliser. Nous vous recommandons de conserver la même version que celle de l'étape précédente et de mettre à jour la version et la configuration dans le cadre d'actions distinctes.
      ii. Sélectionnez Paramètres de configuration facultatifs.
      iii. Entrez la clé JSON `"autoscaling":` et la valeur d'un objet JSON imbriqué avec une clé `"enabled":` et une valeur `true` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`. L'exemple suivant montre que la mise à l'échelle automatique est activée :

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

      iv. (Facultatif) Vous pouvez fournir des valeurs minimales et maximales auxquelles la mise à l'échelle automatique peut ajuster le nombre de pods CoreDNS.

      L'exemple suivant montre que la mise à l'échelle automatique est activée et que toutes les clés facultatives ont des valeurs. Nous recommandons que le nombre minimum de pods CoreDNS soit toujours supérieur à 2 afin de garantir la résilience du service DNS du cluster.

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. Pour appliquer la nouvelle configuration en remplaçant les modules CoreDNS, choisissez Enregistrer les modifications.

   Amazon EKS applique les modifications aux modules complémentaires EKS en déployant le déploiement de Kubernetes pour CoreDNS. Vous pouvez suivre l'état du déploiement dans l'historique des mises à jour du module complémentaire dans AWS Management Console et aveck`ubectl rollout status deployment/coredns --namespace kube-system`.

   `kubectl rollout`possède les commandes suivantes :

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

   Si le déploiement prend trop de temps, Amazon EKS annulera le déploiement et un message indiquant le type de mise à jour de l'extension et le statut Echec sera ajouté à l'historique des mises à jour de l'extension. Pour étudier les problèmes éventuels, commencez par l'historique du déploiement et exécutez `kubectl logs` sur un module CoreDNS pour consulter les journaux de CoreDNS.

5. Si la nouvelle entrée dans l'historique des mises à jour a le statut Successful, le déploiement est terminé et le module complémentaire utilise la nouvelle configuration dans tous les modules CoreDNS. Lorsque vous modifiez le nombre de nœuds et de cœurs de processeur des nœuds du cluster, Amazon EKS adapte le nombre de répliques du déploiement CoreDNS.

Configuration de l'autoscaling CoreDNS dans l'interface de ligne de commande AWS

1. Assurez-vous que votre cluster est égal ou supérieur à la version minimale du cluster.

   Amazon EKS met automatiquement à niveau les clusters entre les versions de plate-forme d'une même version de Kubernetes, et vous ne pouvez pas démarrer ce processus vous-même. Au lieu de cela, vous pouvez mettre à niveau votre cluster vers la prochaine version de Kubernetes, et le cluster sera mis à niveau vers cette version K8s et la dernière version de la plateforme.

   Les nouvelles versions de Kubernetes introduisent parfois des modifications importantes. Par conséquent, nous vous recommandons de tester le comportement de vos applications en utilisant un cluster distinct de la nouvelle version de Kubernetes avant de mettre à jour vos clusters de production.

   Pour mettre à niveau un cluster vers une nouvelle version de Kubernetes, suivez la procédure décrite dans Mettre à jour un cluster existant vers une nouvelle version de Kubernetes.

2. Assurez-vous que vous disposez du module complémentaire EKS pour CoreDNS, et non du déploiement CoreDNS autogéré.

   Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Pour savoir quel type de module complémentaire est installé sur votre cluster, vous pouvez exécuter la commande suivante. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Si une erreur est renvoyée, le type Amazon EKS du module complémentaire n'est pas installé sur votre cluster. Effectuez les étapes restantes de la procédure Créez le module complémentaire CoreDNS Amazon EKS pour remplacer la version autogérée par le module complémentaire Amazon EKS.

3. Assurez-vous que la version de votre module complémentaire EKS pour CoreDNS est identique ou supérieure à la version minimale du module complémentaire EKS.

   Déterminez la version du module complémentaire actuellement installée sur votre cluster. Vous pouvez enregistrer AWS Management Console ou exécuter la commande suivante :

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
d : -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.10.1-eksbuild.13
```

Comparez cette version avec la version minimale du module complémentaire EKS dans la section précédente. Si nécessaire, mettez à niveau le module complémentaire EKS vers une version supérieure en suivant la procédure Mettre à jour le module complémentaire CoreDNS Amazon EKS.

4. Ajoutez la configuration de mise à l'échelle automatique aux paramètres de configuration facultatifs du module complémentaire EKS.

   Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns \
    --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
{"enabled":true}}'
```

   Amazon EKS applique les modifications aux modules complémentaires EKS en déployant le déploiement de Kubernetes pour CoreDNS. Vous pouvez suivre l'état du déploiement dans l'historique des mises à jour du module complémentaire dans AWS Management Console et avec`kubectl rollout status deployment/coredns --namespace kube-system`.

   `kubectl rollout`possède les commandes suivantes :

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Si le déploiement prend trop de temps, Amazon EKS annulera le déploiement et un message indiquant le type de mise à jour de l'extension et le statut Echec sera ajouté à l'historique des mises à jour de l'extension. Pour étudier les problèmes éventuels, commencez par l'historique du déploiement et exécutez `kubectl logs` sur un module CoreDNS pour consulter les journaux de CoreDNS.

5. (Facultatif) Vous pouvez fournir des valeurs minimales et maximales auxquelles la mise à l'échelle automatique peut ajuster le nombre de pods CoreDNS.

   L'exemple suivant montre que la mise à l'échelle automatique est activée et que toutes les clés facultatives ont des valeurs. Nous recommandons que le nombre minimum de pods CoreDNS soit toujours supérieur à 2 afin de garantir la résilience du service DNS du cluster.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true,"minReplicas":2,"maxReplicas":10}}'
   ```

6. Vérifiez l'état de la mise à jour du module complémentaire en exécutant la commande suivante :

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns
   ```

   Si vous voyez cette ligne :`"status": "ACTIVE"`, le déploiement est terminé et le module complémentaire utilise la nouvelle configuration dans tous les modules CoreDNS. Lorsque vous modifiez le nombre de nœuds et de cœurs de processeur des nœuds du cluster, Amazon EKS adapte le nombre de répliques du déploiement CoreDNS.

Surveillez la résolution DNS de Kubernetes avec les métriques CoreDNS

CoreDNS en tant que module complémentaire EKS expose les métriques de CoreDNS sur le port `9153` au format Prometheus dans le service. `kube-dns` Vous pouvez utiliser Prometheus, l'agent CloudWatch Amazon ou tout autre système compatible pour récupérer (collecter) ces statistiques.

Pour un exemple de configuration Scrape compatible à la fois avec Prometheus et avec l'agent, CloudWatch consultez la configuration de CloudWatch l'agent pour Prometheus dans le guide de l'utilisateur Amazon. CloudWatch

Gestion **kube-proxy** dans des clusters Amazon EKS

> **Tip**
> Avec le mode automatique d'Amazon EKS, vous n'avez pas besoin d'installer ou de mettre à niveau des modules complémentaires réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si vous ne connaissez pas la différence entre les types, consultezthe section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Création d'un module complémentaire". Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

Le `kube-proxy` module complémentaire est déployé sur chaque EC2 nœud Amazon de votre cluster Amazon EKS. Il maintient les règles du réseau sur vos nœuds et permet la communication réseau avec vos Pods. Le module complémentaire n'est pas déployé sur les nœuds Fargate de votre cluster. Pour plus d'informations, consultez kube-proxy dans la documentation Kubernetes.

Installation en tant que module complémentaire Amazon EKS

Versions **kube-proxy**

Le tableau suivant répertorie la dernière version du type de module complémentaire Amazon EKS pour chaque version de Kubernetes.

Version de Kubernetes	Version de **kube-proxy**
1,33	v1.33.0-eksbuild.2
1,32	v1.32.6-eksbuild.2
1,31	v1.31.10-eksbuild.2

Version de Kubernetes	Version de **`kube-proxy`**
1,30	v1.30.14-eksbuild.2
1,29	v1.29.15-eksbuild.10
1,28	v1.28.15-eksbuild.25
1,27	v1.27.16-eksbuild.35
1,26	v1.26.15-eksbuild.35

> **Note**
>
> Une version antérieure de la documentation était incorrecte. `kube-proxy`versions `v1.28.5``v1.27.9`, et `v1.26.12` ne sont pas disponibles.
> Si vous gérez vous-même ce module complémentaire, les versions du tableau ne sont peut-être pas les mêmes que les versions autogérées disponibles.

Image de conteneur **`kube-proxy`**

L'image du `kube-proxy` conteneur est basée sur une image de base minimale gérée par Amazon EKS Distro, qui contient un minimum de packages et ne possède pas de shell. Pour plus d'informations, consultez Amazon EKS Distro.

Le tableau suivant répertorie la dernière version d'image de `kube-proxy` conteneur autogérée disponible pour chaque version de cluster Amazon EKS.

Version	kube-proxy
1,33	v1.33.0-minimal-eksbuild.2
1,32	v1.32.6-minimal-eksbuild.2
1,31	v1.31.10-minimal-eksbuild.2
1,30	v1.30.14-minimal-eksbuild.2

Version	kube-proxy
1,29	v1.29.15-minimal-eksbuild.10
1,28	v1.28.15-minimal-eksbuild.25
1,27	v1.27.16-minimal-eksbuild.35
1,26	v1.26.15-minimal-eksbuild.35

Lorsque vous mettez à jour un type de module complémentaire Amazon EKS, vous spécifiez une version valide du module complémentaire Amazon EKS, qui peut ne pas être répertoriée dans ce tableau. Cela est dû au fait que les versions du module complémentaire Amazon EKS ne correspondent pas toujours aux versions d'image de conteneur spécifiées lors de la mise à jour du type autogéré de ce module complémentaire. Lorsque vous mettez à jour le type autogéré de ce module complémentaire, vous spécifiez une version d'image de conteneur valide répertoriée dans ce tableau.

Mettre à jour le module complémentaire autogéré de Kubernetes **`kube-proxy`**

> **Important**
>
> Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si vous ne connaissez pas la différence entre les types, consultezthe section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Création d'un module complémentaire". Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

Prérequis

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route.

Considérations

- `Kube-proxy`sur un cluster Amazon EKS a la même politique de compatibilité et de distorsion que Kubernetes. Découvrez comment vérifier la compatibilité de la version du module complémentaire Amazon EKS avec un cluster.

  1. Vérifiez que le type autogéré du module complémentaire est installé sur votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

     ```
     aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
      addon.addonVersion --output text
     ```

     Si un message d'erreur est renvoyé, le type autogéré du module complémentaire est installé sur votre cluster. Les étapes restantes de cette rubrique concernent la mise à jour du type autogéré du module complémentaire. Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Pour le mettre à jour, utilisez la procédure décrite dans Mettre à jour un module complémentaire Amazon EKS, plutôt que la procédure décrite dans cette rubrique. Si vous ne connaissez pas les différences entre les types de modules complémentaires, consultezthe section called "Modules complémentaires Amazon EKS".

  2. Découvrez quelle version de l'image de conteneur est actuellement installée sur votre cluster.

     ```
     kubectl describe daemonset kube-proxy -n kube-system | grep Image
     ```

     L'exemple qui suit illustre un résultat.

     ```
     Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
     eksbuild.2
     ```

     Dans l'exemple de sortie, *`v1.29.1-eksbuild.2`* la version est-elle installée sur le cluster ?

  3. Mettez à jour le `kube-proxy` module complémentaire en *`region-code`* remplaçant *`602401143452`* et par les valeurs de votre sortie à l'étape précédente. *`v1.30.6-eksbuild.3`*Remplacez-la par la `kube-proxy` version répertoriée dans la dernière version d'image de conteneur kube-proxy autogérée disponible pour chaque tableau des versions de cluster Amazon EKS.

> **Important**
>
> Les manifestes pour chaque type d'image sont différents et ne sont pas compatibles entre les types d'image par défaut et minimaux. Vous devez utiliser le même type d'image que l'image précédente afin que le point d'entrée et les arguments correspondent.

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.6-
eksbuild.3
```

L'exemple qui suit illustre un résultat.

```
daemonset.apps/kube-proxy image updated
```

4. Vérifiez que la nouvelle version est maintenant installée sur votre cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.30.0-eksbuild.3
```

5. Si vous utilisez des `Arm` nœuds `x86` et dans le même cluster et que votre cluster a été déployé avant le 17 août 2020. Ensuite, modifiez votre manifeste `kube-proxy` pour inclure un sélecteur de noeud pour plusieurs architectures matérielles avec la commande suivante. Il s'agit d'une opération ponctuelle. Une fois que vous avez ajouté le sélecteur à votre manifeste, vous n'avez pas besoin de l'ajouter chaque fois que vous mettez à jour le module complémentaire. Si votre cluster a été déployé à partir du 17 août 2020, `kube-proxy` est déjà compatible avec les architectures multiples.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Ajoutez le sélecteur de nœuds suivant au fichier dans l'éditeur, puis enregistrez le fichier. Pour obtenir un exemple d'inclusion de ce texte dans l'éditeur, consultez le fichier manifeste CNI

sur GitHub. Cela permet à Kubernetes d'extraire l'image matérielle correcte en fonction de l'architecture matérielle du nœud.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Si votre cluster a été créé à l'origine avec une version de Kubernetes `1.14` ou ultérieure, vous pouvez ignorer cette étape car elle l'inclut `kube-proxy` déjà. `Affinity Rule` Si vous avez initialement créé un cluster Amazon EKS avec la version de Kubernetes `1.13` ou une version antérieure et que vous avez l'intention d'utiliser des nœuds Fargate dans votre cluster, modifiez votre `kube-proxy` manifeste pour inclure une `NodeAffinity` règle interdisant la planification des pods `kube-proxy` sur les nœuds Fargate. Il s'agit d'une modification unique. Une fois que vous avez ajouté le `Affinity Rule` à votre manifeste, vous n'avez pas besoin de l'ajouter à chaque mise à jour du module complémentaire. Modifiez votre `kube-proxy` DaemonSet.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Ajoutez ce qui suit `Affinity Rule` à la DaemonSet `spec` section du fichier dans l'éditeur, puis enregistrez le fichier. Pour obtenir un exemple d'inclusion de ce texte dans l'éditeur, consultez le fichier manifeste CNI sur GitHub.

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

Découvrez comment déployer des charges de travail et des modules complémentaires sur Amazon EKS

Vos charges de travail sont déployées dans des conteneurs, qui sont déployés dans des pods dans Kubernetes. Un pod comprend un ou plusieurs conteneurs. Généralement, un ou plusieurs pods fournissant le même service sont déployés dans un service Kubernetes. Une fois que vous avez déployé plusieurs pods fournissant le même service, vous pouvez :

- Afficher des informations sur les applications s'exécutant sur chacun de vos clusters à l'aide de AWS Management Console.
- Augmentez ou diminuez verticalement les pods avec le Kubernetes Vertical Pod Autoscaler.
- Augmentez ou diminuez horizontalement le nombre de pods nécessaires pour répondre à la demande avec le Kubernetes Horizontal Pod Autoscaler.
- Créez un équilibreur de charge réseau externe (pour les pods accessibles à Internet) ou interne (pour les pods privés) afin d'équilibrer le trafic réseau entre les pods. L'équilibreur de charge achemine le trafic à la couche 4 du modèle OSI.
- Créez un Application Load Balancer pour équilibrer le trafic des applications entre les pods. L'Application Load Balancer achemine le trafic à la couche 7 du modèle OSI.
- Si vous utilisez Kubernetes pour la première fois, cette rubrique vous aide à déployer un exemple d'application.
- Vous pouvez restreindre les adresses IP qui peuvent être affectées à un service avec `externalIPs`.

Déployer un exemple d'application sous Linux

Dans cette rubrique, vous allez déployer un exemple d'application sur votre cluster sur des nœuds Linux.

Prérequis

- Un cluster Kubernetes existant avec au moins un nœud. Si vous ne possédez pas de cluster Amazon EKS existant, vous pouvez en déployer un à l'aide de l'un des guides fournis dansMise en route.

- `Kubectl` installé sur votre ordinateur. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".
- `Kubectl` configuré pour communiquer avec votre cluster. Pour de plus amples informations, veuillez consulter the section called "Cluster d'accès avec kubectl".
- Si vous prévoyez de déployer votre charge de travail d'exemple sur Fargate, vous devez disposer d'un profil Fargate existant qui inclut le même espace de noms créé dans ce didacticiel, à savoir `eks-sample-app`, à moins que vous ne changiez le nom. Si vous avez créé un cluster avec l'un des guidesMise en route, vous devrez créer un nouveau profil ou ajouter l'espace de noms à votre profil existant, car le profil créé dans les guides de démarrage ne spécifie pas l'espace de noms utilisé dans ce didacticiel. Votre VPC doit également disposer d'au moins un sous-réseau privé.

Bien que de nombreuses variables soient modifiables dans les étapes suivantes, nous vous recommandons de ne modifier les valeurs des variables qu'aux endroits spécifiés. Une fois que vous aurez mieux compris les pods, les déploiements et les services Kubernetes, vous pourrez essayer de modifier d'autres valeurs.

Création d'un espace de noms

Un espace de noms vous permet de regrouper des ressources dans Kubernetes. Pour plus d'informations, consultez Espaces de noms dans la documentation Kubernetes. Si vous prévoyez de déployer votre exemple d'application pour simplifier la gestion des calculs avec AWS Fargate, assurez-vous que la valeur `namespace` indiquée dans votre section Définir les pods qui utilisent AWS Fargate au lancement est. `eks-sample-app`

```
kubectl create namespace eks-sample-app
```

Création d'un déploiement Kubernetes

Créer un déploiement Kubernetes. Cet exemple de déploiement extrait une image de conteneur d'un référentiel public et déploie trois répliques (pods individuels) de celle-ci sur votre cluster. Pour en savoir plus, consultez Déploiements dans la documentation Kubernetes.

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-deployment.yaml`. Les conteneurs de l'exemple d'application n'utilisent pas le stockage réseau, mais certaines applications en ont peut-être besoin. Pour de plus amples informations, veuillez consulter Stockage des données des applications.

- Le `amd64` ou `arm64` `values` sous la clé `kubernetes.io/arch` signifient que l'application peut être déployée sur l'une ou l'autre architecture matérielle (si vous avez les deux dans votre cluster). Cela est possible car cette image est une image multi-architecture, mais toutes ne le sont pas. Vous pouvez déterminer l'architecture matérielle sur laquelle l'image est prise en charge en consultant les détails de l'image dans le référentiel d'où vous l'extrayez. Lorsque vous déployez des images qui ne prennent pas en charge un type d'architecture matérielle ou sur lesquelles vous ne souhaitez pas que l'image soit déployée, supprimez ce type du manifeste. Pour plus d'informations, consultez Étiquettes, annotations et rejets connus dans la documentation Kubernetes.
- Le `kubernetes.io/os: linux` `nodeSelector` signifie que si vous possédez des nœuds Linux et Windows (par exemple) dans votre cluster, l'image ne sera déployée que sur des nœuds Linux. Pour plus d'informations, consultez Étiquettes, annotations et rejets connus dans la documentation Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
```

```
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

2. Appliquer le manifeste de déploiement à votre cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Créer un service

Un service vous permet d'accéder à toutes les réplicas via une seule adresse IP ou un seul nom. Pour plus d'informations, consultez Service dans la documentation Kubernetes. Bien que cela ne soit pas implémenté dans l'exemple d'application, si certaines applications doivent interagir avec d'autres AWS services, nous vous recommandons de créer des comptes de service Kubernetes pour vos pods et de les associer à AWS des comptes IAM. En spécifiant des comptes de service, vos pods ne disposent que des autorisations minimales que vous spécifiez pour qu'ils puissent interagir avec d'autres services. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA".

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-service.yaml`. Kubernetes attribue au service sa propre adresse IP qui n'est accessible que depuis l'intérieur du cluster. Pour accéder au service depuis l'extérieur de votre cluster, déployez l'AWS Load Balancer Controller pour équilibrer la charge du trafic d'application ou de réseau vers le service.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
```

```
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Appliquer le manifeste du service à votre cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Réviser les ressources créées

1. Affichez toutes les ressources présentes dans l'espace de noms `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

L'exemple qui suit illustre un résultat.

```
NAME                                                    READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-linux-service   ClusterIP    10.100.74.8     <none>        80/TCP
   32m

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment 3/3     3            3           27m

NAME                                                       DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
 27m
```

Dans la sortie, vous voyez le service et le déploiement spécifiés dans l'exemple de manifestes déployés dans les étapes précédentes. Vous voyez aussi trois pods. Ceci est dû au fait que 3 `replicas` ont été spécifiés dans l'exemple de manifeste. Pour plus d'informations sur les pods, consultez pods dans la documentation Kubernetes. Kubernetes crée automatiquement la

`replicaset` ressource, même si elle n'est pas spécifiée dans les exemples de manifestes. Pour plus d'informations`ReplicaSets`, consultez la documentation ReplicaSetde Kubernetes.

> **Note**
>
> Kubernetes conserve le nombre de réplicas spécifié dans le manifeste. S'il s'agissait d'un déploiement de production et que vous souhaitiez que Kubernetes redimensionne horizontalement le nombre de répliques ou redimensionne verticalement les ressources de calcul pour les pods, utilisez les déploiements de pods avec Horizontal Pod Autoscaler et Adjust les ressources des pods avec Vertical Pod Autoscaler pour ce faire.

2. Affichez les détails du service déployé.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

L'exemple qui suit illustre un résultat.

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Dans la sortie précédente, la valeur pour `IP:` est une adresse IP unique accessible depuis n'importe quel nœud ou pod du cluster, mais pas depuis l'extérieur du cluster. Les valeurs pour `Endpoints` sont les adresses IP attribuées depuis votre VPC aux pods qui font partie du service.

3. Visualisez les détails de l'un des pods listés en sortie lorsque vous avez visualisé l'espace de noms dans une étape précédente. *`776d8f8fd8-78w66`*Remplacez-le par la valeur renvoyée pour l'un de vos pods.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Exemple de sortie abrégé

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
 ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Dans la sortie précédente, la valeur pour `IP:` est une adresse IP unique attribuée au Pod à partir du bloc CIDR attribué au sous-réseau dans lequel se trouve le nœud. Si vous préférez attribuer aux pods des adresses IP à partir de différents blocs d'adresse CIDR, vous pouvez modifier le comportement par défaut. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée". Vous pouvez également voir que le planificateur Kubernetes a programmé le pod sur le `Node` avec l'adresse IP *`192.168.45.132`*.

> **Tip**
>
> Plutôt que d'utiliser la ligne de commande, vous pouvez consulter de nombreux détails sur les pods, les services, les déploiements et les autres ressources Kubernetes dans le. AWS Management Console Pour de plus amples informations, veuillez consulter the section called "Accès aux ressources du cluster".

Exécuter un shell sur un Pod

1. Exécutez un shell sur le pod que vous avez décrit à l'étape précédente, en le *65b7669776-m6qxz* remplaçant par l'identifiant de l'un de vos pods.

```
kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /
bin/bash
```

2. À partir du shell Pod, visualisez la sortie du serveur Web qui a été installé lors de votre déploiement lors d'une étape précédente. Vous devez uniquement spécifier le nom du service. Il est résolu à l'adresse IP du service par CoreDNS, qui est déployé par défaut avec un cluster Amazon EKS.

```
curl eks-sample-linux-service
```

   L'exemple qui suit illustre un résultat.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

3. À partir du shell du pod, affichez le serveur DNS pour le pod.

```
cat /etc/resolv.conf
```

   L'exemple qui suit illustre un résultat.

```
nameserver 10.100.0.10
```

```
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.compute.internal
options ndots:5
```

Dans la sortie précédente, `10.100.0.10` est automatiquement affecté en tant que `nameserver` pour tous les pods déployés dans le cluster.

4. Déconnectez-vous du pod en tapant `exit`.
5. Une fois que vous avez terminé d'utiliser l'exemple d'application, vous pouvez supprimer l'espace de noms, le service et le déploiement de l'exemple à l'aide de la commande suivante.

```
kubectl delete namespace eks-sample-app
```

Étapes suivantes

Après avoir déployé l'exemple d'application, vous pouvez essayer certains des exercices suivants :

- Acheminez le trafic d'applications et le trafic HTTP avec des équilibreurs de charge d'application
- Acheminez le trafic TCP et UDP avec des équilibreurs de charge réseau

Déployer un exemple d'application sous Windows

Dans cette rubrique, vous allez déployer un exemple d'application sur votre cluster sur des nœuds Windows.

Prérequis

- Un cluster Kubernetes existant avec au moins un nœud. Si vous ne possédez pas de cluster Amazon EKS existant, vous pouvez en déployer un à l'aide de l'un des guides fournis dansMise en route. Le support Windows doit être activé pour votre cluster et au moins un nœud Amazon EC2 Windows.
- `Kubectl` installé sur votre ordinateur. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".
- `Kubectl` configuré pour communiquer avec votre cluster. Pour de plus amples informations, veuillez consulter the section called "Cluster d'accès avec kubectl".
- Si vous prévoyez de déployer votre charge de travail d'exemple sur Fargate, vous devez disposer d'un profil Fargate existant qui inclut le même espace de noms créé dans ce didacticiel, à savoir

`eks-sample-app`, à moins que vous ne changiez le nom. Si vous avez créé un cluster avec l'un des guidesMise en route, vous devrez créer un nouveau profil ou ajouter l'espace de noms à votre profil existant, car le profil créé dans les guides de démarrage ne spécifie pas l'espace de noms utilisé dans ce didacticiel. Votre VPC doit également disposer d'au moins un sous-réseau privé.

Bien que de nombreuses variables soient modifiables dans les étapes suivantes, nous vous recommandons de ne modifier les valeurs des variables qu'aux endroits spécifiés. Une fois que vous aurez mieux compris les pods, les déploiements et les services Kubernetes, vous pourrez essayer de modifier d'autres valeurs.

Création d'un espace de noms

Un espace de noms vous permet de regrouper des ressources dans Kubernetes. Pour plus d'informations, consultez Espaces de noms dans la documentation Kubernetes. Si vous prévoyez de déployer votre exemple d'application pour simplifier la gestion des calculs avec AWS Fargate, assurez-vous que la valeur `namespace` indiquée dans votre section Définir les pods qui utilisent AWS Fargate au lancement est. `eks-sample-app`

```
kubectl create namespace eks-sample-app
```

Création d'un déploiement Kubernetes

Cet exemple de déploiement extrait une image de conteneur d'un référentiel public et déploie trois répliques (pods individuels) de celle-ci sur votre cluster. Pour en savoir plus, consultez Déploiements dans la documentation Kubernetes.

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-deployment.yaml`. Les conteneurs de l'exemple d'application n'utilisent pas le stockage réseau, mais certaines applications en ont peut-être besoin. Pour de plus amples informations, veuillez consulter Stockage des données des applications.
   - Le `kubernetes.io/os: windows` `nodeSelector` signifie que si vous possédez des nœuds Windows et Linux (par exemple) dans votre cluster, l'image ne sera déployée que sur des nœuds Windows. Pour plus d'informations, consultez Étiquettes, annotations et rejets connus dans la documentation Kubernetes.

     ```
     apiVersion: apps/v1
     kind: Deployment
     ```

```
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\\wwwroot\
\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

2. Appliquer le manifeste de déploiement à votre cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Créer un service

Un service vous permet d'accéder à toutes les réplicas via une seule adresse IP ou un seul nom. Pour plus d'informations, consultez Service dans la documentation Kubernetes. Bien que cela ne soit pas implémenté dans l'exemple d'application, si certaines applications doivent interagir avec d'autres AWS services, nous vous recommandons de créer des comptes de service Kubernetes pour vos pods et de les associer à AWS des comptes IAM. En spécifiant des comptes de service, vos pods ne disposent que des autorisations minimales que vous spécifiez pour qu'ils puissent interagir avec d'autres services. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA".

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-service.yaml`. Kubernetes attribue au service sa propre adresse IP qui n'est accessible que depuis l'intérieur du cluster. Pour accéder au service depuis l'extérieur de votre cluster, déployez l'AWS Load Balancer Controller pour équilibrer la charge du trafic d'application ou de réseau vers le service.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Appliquer le manifeste du service à votre cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Réviser les ressources créées

1. Affichez toutes les ressources présentes dans l'espace de noms `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

L'exemple qui suit illustre un résultat.

```
NAME                                                    READY   STATUS    RESTARTS   AGE
pod/eks-sample-windows-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-windows-service   ClusterIP   10.100.74.8     <none>        80/
TCP    32m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-windows-deployment 3/3     3            3           27m

NAME                                                       DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-windows-deployment-776d8f8fd8    3         3         3
   27m
```

Dans la sortie, vous voyez le service et le déploiement spécifiés dans l'exemple de manifestes déployés dans les étapes précédentes. Vous voyez aussi trois pods. Ceci est dû au fait que 3 `replicas` ont été spécifiés dans l'exemple de manifeste. Pour plus d'informations sur les pods, consultez pods dans la documentation Kubernetes. Kubernetes crée automatiquement la `replicaset` ressource, même si elle n'est pas spécifiée dans les exemples de manifestes. Pour plus d'informations`ReplicaSets`, consultez la documentation ReplicaSetde Kubernetes.

> **Note**
>
> Kubernetes conserve le nombre de réplicas spécifié dans le manifeste. S'il s'agissait d'un déploiement de production et que vous souhaitiez que Kubernetes redimensionne horizontalement le nombre de répliques ou redimensionne verticalement les ressources de

calcul pour les pods, utilisez les déploiements de pods avec Horizontal Pod Autoscaler et Adjust les ressources des pods avec Vertical Pod Autoscaler pour ce faire.

2. Affichez les détails du service déployé.

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

L'exemple qui suit illustre un résultat.

```
Name:              eks-sample-windows-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-windows-app
Annotations:       <none>
Selector:          app=eks-sample-windows-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Dans la sortie précédente, la valeur pour `IP:` est une adresse IP unique accessible depuis n'importe quel nœud ou pod du cluster, mais pas depuis l'extérieur du cluster. Les valeurs pour `Endpoints` sont les adresses IP attribuées depuis votre VPC aux pods qui font partie du service.

3. Visualisez les détails de l'un des pods listés en sortie lorsque vous avez visualisé l'espace de noms dans une étape précédente. *`776d8f8fd8-78w66`*Remplacez-le par la valeur renvoyée pour l'un de vos pods.

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

Exemple de sortie abrégé

```
Name:         eks-sample-windows-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
```

```
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-windows-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-windows-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Dans la sortie précédente, la valeur pour `IP:` est une adresse IP unique attribuée au Pod à partir du bloc CIDR attribué au sous-réseau dans lequel se trouve le nœud. Si vous préférez attribuer aux pods des adresses IP à partir de différents blocs d'adresse CIDR, vous pouvez modifier le comportement par défaut. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée". Vous pouvez également voir que le planificateur Kubernetes a programmé le pod sur le `Node` avec l'adresse IP *`192.168.45.132`*.

> **Tip**
>
> Plutôt que d'utiliser la ligne de commande, vous pouvez consulter de nombreux détails sur les pods, les services, les déploiements et les autres ressources Kubernetes dans le. AWS Management Console Pour de plus amples informations, veuillez consulter the section called "Accès aux ressources du cluster".

Exécuter un shell sur un Pod

1. Exécutez un shell sur le pod que vous avez décrit à l'étape précédente, en le *65b7669776-m6qxz* remplaçant par l'identifiant de l'un de vos pods.

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app --
 powershell.exe
```

2. À partir du shell Pod, visualisez la sortie du serveur Web qui a été installé lors de votre déploiement lors d'une étape précédente. Vous devez uniquement spécifier le nom du service. Il est résolu à l'adresse IP du service par CoreDNS, qui est déployé par défaut avec un cluster Amazon EKS.

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

L'exemple qui suit illustre un résultat.

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > < H
 1 > H e l l o
                    E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l >
```

3. À partir du shell du pod, affichez le serveur DNS pour le pod.

```
Get-NetIPConfiguration
```

Sortie abrégée

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

Dans la sortie précédente, `10.100.0.10` est automatiquement affecté en tant que serveur DNS pour tous les pods déployés dans le cluster.

4. Déconnectez-vous du pod en tapant `exit`.

5. Une fois que vous avez terminé d'utiliser l'exemple d'application, vous pouvez supprimer l'espace de noms, le service et le déploiement de l'exemple à l'aide de la commande suivante.

```
kubectl delete namespace eks-sample-app
```

Étapes suivantes

Après avoir déployé l'exemple d'application, vous pouvez essayer certains des exercices suivants :

- Acheminez le trafic d'applications et le trafic HTTP avec des équilibreurs de charge d'application
- Acheminez le trafic TCP et UDP avec des équilibreurs de charge réseau

Ajustez les ressources du pod avec Vertical Pod Autoscaler

Le Kubernetes Vertical Pod Autoscaler ajuste automatiquement les réserves de processeur et de mémoire pour vos pods afin de vous aider à « dimensionner correctement » vos applications. Cet ajustement peut améliorer l'utilisation des ressources du cluster et libérer du processeur et de la mémoire pour d'autres pods. Cette rubrique vous aide à déployer le VPA (Vertical Pod Autoscaler) sur votre cluster et à vérifier qu'il fonctionne.

- Vous disposez d'un cluster Amazon EKS existant. Si ce n'est pas le cas, voyezMise en route.
- Le serveur de métriques Kubernetes est installé. Pour de plus amples informations, veuillez consulter the section called “Serveur de métriques”.
- Vous utilisez un client `kubectl` qui est configuré pour communiquer avec votre cluster Amazon EKS.
- OpenSSL `1.1.1` ou version ultérieure installé sur votre appareil.

Déploiement du VPA (Vertical Pod Autoscaler)

Dans cette section, vous déployez le Vertical Pod Autoscaler sur votre cluster.

1. Ouvrez une fenêtre de terminal et accédez à un répertoire dans lequel vous souhaitez télécharger le code source du VPA (Vertical Pod Autoscaler).
2. Clonez le référentiel GitHub kubernetes/autoscaler.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Passez au répertoire `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Facultatif) Si vous avez déjà déployé une autre version du VPA (Vertical Pod Autoscaler), supprimez-la avec la commande suivante.

```
./hack/vpa-down.sh
```

5. Si vos nœuds ne disposent pas d'un accès Internet au registre des `registry.k8s.io` conteneurs, vous devez extraire les images suivantes et les transférer vers votre propre référentiel privé. Pour plus d'informations sur la manière d'extraire les images et de les transférer vers votre propre référentiel privé, consultez the section called "Copier une image dans un référentiel".

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

   Si vous transférez les images vers un référentiel Amazon ECR privé, remplacez-les `registry.k8s.io` dans les manifestes par votre registre. Remplacez *111122223333* par votre ID de compte. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster. Les commandes suivantes supposent que vous avez nommé votre référentiel de la même manière que le référentiel du manifeste. Si vous avez donné un autre nom à votre dépôt, vous devrez également le modifier.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/updater-deployment.yaml
```

6. Déployez le VPA (Vertical Pod Autoscaler) sur votre cluster avec la commande suivante.

```
./hack/vpa-up.sh
```

7. Vérifiez que les pods Vertical Pod Autoscaler ont été créés avec succès.

```
kubectl get pods -n kube-system
```

L'exemple qui suit illustre un résultat.

```
NAME                                       READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w             1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd  1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl           1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d               1/1     Running   0          8s
```

Test de l'installation de votre VPA (Vertical Pod Autoscaler)

Dans cette section, vous déployez un exemple d'application pour vérifier que le Vertical Pod Autoscaler fonctionne.

1. Déployez l'exemple `hamster.yaml` Vertical Pod Autoscaler avec la commande suivante.

```
kubectl apply -f examples/hamster.yaml
```

2. Obtenez les Pods à partir de l'`hamster`exemple d'application.

```
kubectl get pods -l app=hamster
```

L'exemple qui suit illustre un résultat.

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Décrivez l'un des pods pour le consulter `cpu` et `memory` réserver. *`c7d89d6db-rglf5`*Remplacez-le par l'un des résultats IDs renvoyés à l'étape précédente.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

L'exemple qui suit illustre un résultat.

```
[...]
Containers:
```

```
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

Vous pouvez voir que le Pod d'origine réserve 100 millicpu de processeur et 50 mégaoctets de mémoire. Pour cet exemple d'application, 100 millicpu sont inférieurs à ce dont le Pod a besoin pour fonctionner, il est donc limité par le processeur. Il réserve également beaucoup moins de mémoire qu'il n'en a besoin. Le `vpa-recommender` déploiement de Vertical Pod Autoscaler analyse les hamsters Pods pour déterminer si les exigences en matière de processeur et de mémoire sont appropriées. Si des ajustements sont nécessaires, les pods sont `vpa-updater` relancés avec des valeurs mises à jour.

4. Attendez le lancement `vpa-updater` d'un nouveau hamster Pods. Cela devrait prendre une minute ou deux. Vous pouvez surveiller les Pods à l'aide de la commande suivante.

> **Note**
>
> Si vous n'êtes pas sûr qu'un nouveau pod ait été lancé, comparez les noms des pods avec votre liste précédente. Lorsque le nouveau Pod sera lancé, vous verrez un nouveau nom de Pod.

```
kubectl get --watch Pods -l app=hamster
```

5. Lorsqu'un nouveau hamster Pods est démarré, décrivez-le et consultez les réservations de processeur et de mémoire mises à jour.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

L'exemple qui suit illustre un résultat.

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

Dans la sortie précédente, vous pouvez voir que la réservation de l'`cpu` est passée à 587 millicpu, ce qui représente plus de cinq fois la valeur d'origine. La `memory` a augmenté jusqu'à 262 144 kilo-octets, soit environ 250 mébioctets ou cinq fois la valeur d'origine. Ce Pod manquait de ressources, et le Vertical Pod Autoscaler a corrigé l'estimation avec une valeur beaucoup plus appropriée.

6. Décrivez la ressource `hamster-vpa` pour afficher la nouvelle recommandation.

```
kubectl describe vpa/hamster-vpa
```

L'exemple qui suit illustre un résultat.

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":{},"name":"hamster-
vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
```

```
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. Lorsque vous avez terminé de tester votre exemple d'application, supprimez-le avec la commande suivante.

```
kubectl delete -f examples/hamster.yaml
```

Faites évoluer les déploiements de pods avec Horizontal Pod Autoscaler

Le Kubernetes Horizontal Pod Autoscaler adapte automatiquement le nombre de pods dans un déploiement, un contrôleur de réplication ou un ensemble de répliques en fonction de l'utilisation du processeur de cette ressource. Cela peut permettre à vos applications de monter en puissance pour répondre à une demande accrue ou de baisser en puissance lorsque les ressources ne sont pas nécessaires, libérant ainsi vos nœuds pour d'autres applications. Lorsque vous définissez un pourcentage cible d'utilisation du processeur, Horizontal Pod Autoscaler met à l'échelle votre application pour essayer d'atteindre cet objectif.

Horizontal Pod Autoscaler est une ressource API standard dans Kubernetes qui nécessite simplement qu'une source de métriques (telle que le serveur de métriques Kubernetes) soit installée sur votre cluster Amazon EKS pour fonctionner. Vous n'avez pas besoin de déployer ou d'installer Horizontal Pod Autoscaler sur votre cluster pour mettre vos applications à l'échelle. Pour plus d'informations, consultez Horizontal Pod Autoscaler dans la documentation Kubernetes.

Utilisez cette rubrique pour préparer Horizontal Pod Autoscaler pour votre cluster Amazon EKS et pour vérifier qu'il fonctionne avec un exemple d'application.

> **Note**
>
> Cette rubrique est basée sur la procédure pas à pas de l'autoscaler Horizontal Pod présentée dans la documentation de Kubernetes.

- Vous disposez d'un cluster Amazon EKS existant. Si ce n'est pas le cas, voyezMise en route.

- Le serveur de métriques Kubernetes est installé. Pour de plus amples informations, veuillez consulter the section called "Serveur de métriques".
- Vous utilisez un client `kubectl` qui est configuré pour communiquer avec votre cluster Amazon EKS.

Exécution d'une application de test du Horizontal Pod Autoscaler

Dans cette section, vous déployez un exemple d'application pour vérifier que Horizontal Pod Autoscale) fonctionne.

> **Note**
>
> Cet exemple est basé sur la procédure pas à pas de l'autoscaler Horizontal Pod présentée dans la documentation de Kubernetes.

1. Déployez une application simple de serveur web Apache avec la commande suivante.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   Ce Pod de serveur Web Apache est doté d'une limite de processeur de 500 millicpu et il fonctionne sur le port 80.
2. Créez une ressource Horizontal Pod Autoscaler pour le déploiement `php-apache`.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   Cette commande crée un autoscaler qui cible 50 % d'utilisation du processeur pour le déploiement, avec un minimum d'un pod et un maximum de dix pods. Lorsque la charge moyenne du processeur est inférieure à 50 %, l'autoscaler essaie de réduire le nombre de pods dans le déploiement à un minimum d'un. Lorsque la charge est supérieure à 50 %, l'autoscaler essaie d'augmenter le nombre de pods dans le déploiement, jusqu'à un maximum de dix. Pour plus d'informations, voir Comment HorizontalPodAutoscaler fonctionne un dans la documentation de Kubernetes.
3. Décrivez l'autoscaler avec la commande suivante pour en afficher les détails.

   ```
   kubectl get hpa
   ```

L'exemple qui suit illustre un résultat.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          51s
```

Comme vous pouvez le constater, la charge actuelle du processeur est due `0%` au fait qu'il n'y a pas encore de charge sur le serveur. Le nombre de pods a déjà atteint sa limite la plus basse (un), il ne peut donc pas être augmenté.

4. Créez une charge pour le serveur Web en exécutant un conteneur.

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. Pour observer l'augmentation du déploiement, exécutez régulièrement la commande suivante dans un terminal distinct de celui dans lequel vous avez exécuté l'étape précédente.

```
kubectl get hpa php-apache
```

L'exemple qui suit illustre un résultat.

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    250%/50%   1         10        5          4m44s
```

L'augmentation du nombre de réplicas peut prendre plus d'une minute. Tant que le pourcentage de CPU réel est supérieur au pourcentage cible, le nombre de réplicas augmente jusqu'à 10. Dans ce cas, c'est le cas`250%`, donc le nombre `REPLICAS` continue d'augmenter.

> **Note**
>
> L'atteinte du nombre de réplicas maximum peut prendre un certain temps. Si seulement 6 répliques, par exemple, sont nécessaires pour que la charge du processeur reste égale ou inférieure à 50 %, la charge ne dépassera pas 6 répliques.

6. Arrêtez la charge. Dans la fenêtre du terminal dans laquelle vous générez le chargement, arrêtez le chargement en maintenant les `Ctrl+C` touches enfoncées. Vous pouvez voir les répliques

revenir à 1 en exécutant à nouveau la commande suivante dans le terminal dans lequel vous observez la mise à l'échelle.

```
kubectl get hpa
```

L'exemple qui suit illustre un résultat.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          25m
```

> **Note**
>
> Le délai par défaut pour la réduction d'échelle est de cinq minutes. Par conséquent, un certain temps s'écoule avant de voir le nombre de réplicas atteindre à nouveau 1, même si le pourcentage de CPU actuel est de 0 %. Le délai est modifiable. Pour plus d'informations, consultez Horizontal Pod Autoscaler dans la documentation Kubernetes.

7. Lorsque vous avez terminé de tester votre exemple d'application, supprimez les ressources `php-apache`.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Acheminez le trafic TCP et UDP avec des équilibreurs de charge réseau

> **Note**
>
> Nouveau : le mode automatique d'Amazon EKS automatise les tâches de routine pour l'équilibrage de charge. Pour plus d'informations, consultez :
>
> - the section called "Déployer un équilibreur de charge"
> - the section called "Créer un service"

Le trafic réseau est équilibré en charge au niveau `L4` du modèle OSI. Pour équilibrer la charge du trafic des applications sur`L7`, vous déployez un Kubernetes`ingress`, qui fournit un Application AWS Load Balancer. Pour de plus amples informations, veuillez consulter the section called "Répartition de la charge des applications". Pour en savoir plus sur les différences entre les deux types d'équilibrage de charge, consultez les fonctionnalités d'Elastic Load Balancing sur le AWS site Web.

Lorsque vous créez un Kubernetes `Service` de type Kubernetes`LoadBalancer`, le contrôleur d'équilibrage de charge du fournisseur de AWS cloud crée des équilibreurs de charge AWS classiques par défaut, mais peut également créer des équilibreurs de charge réseau. AWS Ce contrôleur ne recevra que des corrections de bugs critiques à l'avenir. Pour plus d'informations sur l'utilisation de l'équilibreur de charge du fournisseur de AWS cloud, consultez le contrôleur d'équilibrage de charge du fournisseur de AWS cloud dans la documentation de Kubernetes. Son utilisation n'est pas abordée dans cette rubrique.

Nous vous recommandons d'utiliser la version `2.7.2` ou une version ultérieure du AWS Load Balancer Controller au lieu du contrôleur d'équilibrage de charge du fournisseur de AWS cloud. Le AWS Load Balancer Controller crée des équilibreurs de charge AWS réseau, mais pas des équilibreurs de charge AWS classiques. Le reste de cette rubrique traite de l'utilisation du AWS Load Balancer Controller.

Un AWS Network Load Balancer peut équilibrer la charge du trafic réseau vers les pods déployés sur Amazon EC2 IP et les cibles d'instance, vers les cibles IP Fargate ou vers AWS les nœuds hybrides Amazon EKS en tant que cibles IP. Pour plus d'informations, consultez AWS Load Balancer Controller activé. GitHub

Prérequis

Avant de pouvoir équilibrer la charge du trafic réseau à l'aide du AWS Load Balancer Controller, vous devez satisfaire aux exigences suivantes.

- Disposer d'un cluster. Si vous n'avez pas de cluster existant, consultezMise en route. Si vous devez mettre à jour la version d'un cluster existant, consultez the section called "Mettre à jour la version de Kubernetes".
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.

- Au moins un sous-réseau. Si plusieurs sous-réseaux identifiés sont trouvés dans une zone de disponibilité, le contrôleur choisit le sous-réseau dont l'ID vient en premier dans l'ordre lexicographique. Le sous-réseau doit disposer au moins huit adresses IP disponibles.
- Si vous utilisez la version du AWS Load Balancer Controller `2.1.1` ou une version antérieure, les sous-réseaux doivent être balisés comme suit. Si vous utilisez la version `2.1.2` ou une version ultérieure, cette identification est facultative. Vous souhaiterez peut-être baliser un sous-réseau si plusieurs clusters s'exécutent dans le même VPC, ou si AWS plusieurs services partagent des sous-réseaux dans un VPC, et si vous souhaitez mieux contrôler l'endroit où les équilibreurs de charge sont fournis pour chaque cluster. Si vous spécifiez explicitement un sous-réseau IDs sous forme d'annotation sur un objet de service, Kubernetes et le Load Balancer Controller utilisent directement ces sous-réseaux pour créer l'équilibreur de AWS charge. Le balisage des sous-réseaux n'est pas nécessaire si vous choisissez d'utiliser cette méthode pour configurer des équilibreurs de charge et vous pouvez ignorer les exigences de balisage des sous-réseaux privés et publics suivantes. Remplacez *`my-cluster`* par le nom de votre cluster.
  - Clé : `kubernetes.io/cluster/<my-cluster>`
  - Valeur : `shared` ou `owned`
- Vos sous-réseaux publics et privés doivent répondre aux exigences suivantes, sauf si vous le spécifiez explicitement sous IDs forme d'annotation sur un service ou un objet d'entrée. Si vous configurez des équilibreurs de charge en spécifiant explicitement un sous-réseau IDs sous forme d'annotation sur un service ou un objet d'entrée, Kubernetes et le Load AWS Balancer Controller utilisent ces sous-réseaux directement pour créer l'équilibreur de charge et les balises suivantes ne sont pas requises.
  - Sous-réseaux privés : doivent être étiquetés dans le format suivant. Cela permet à Kubernetes et au Load Balancer Controller de savoir que les sous-réseaux peuvent être utilisés pour les équilibreurs de AWS charge internes. Si vous utilisez `eksctl` un AWS AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de VPC Amazon EKS AWS AWS CloudFormation , consultez the section called “Création d'un VPC”.
    - Clé : `kubernetes.io/role/internal-elb`
    - Valeur : `1`
  - Sous-réseau publics : doivent être étiquetés dans le format suivant. Kubernetes sait ainsi qu'il ne faut utiliser que ces sous-réseaux pour les équilibreurs de charge externes au lieu de choisir un sous-réseau public dans chaque zone de disponibilité (en fonction de l'ordre lexicographique du sous-réseau). IDs Si vous utilisez `eksctl` un AWS CloudFormation modèle Amazon EKS pour

créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de AWS CloudFormation VPC Amazon EKS, consultez. the section called "Création d'un VPC"

- Clé : `kubernetes.io/role/elb`
- Valeur : `1`

Si les balises de rôle de sous-réseau ne sont pas explicitement ajoutées, le contrôleur de service Kubernetes examine la table de routage des sous-réseaux VPC de votre cluster pour déterminer s'il s'agit d'un sous-réseau privé ou public. Nous vous recommandons de ne pas vous fier à ce comportement et d'ajouter explicitement les balises de rôle privées ou publiques. Le AWS Load Balancer Controller n'examine pas les tables de routage et nécessite la présence des balises privées et publiques pour une détection automatique réussie.

Considérations

- La configuration de votre équilibreur de charge est contrôlée par des annotations qui sont ajoutées au manifeste de votre service. Les annotations de service sont différentes lorsque vous utilisez le AWS Load Balancer Controller et lorsque vous utilisez AWS le contrôleur d'équilibrage de charge du fournisseur de cloud. Assurez-vous de consulter les annotations relatives au AWS Load Balancer Controller avant de déployer des services.
- Lorsque vous utilisez le plug-in Amazon VPC CNI pour Kubernetes, le Load AWS Balancer Controller peut équilibrer la charge entre EC2 Amazon IP ou les cibles d'instance et les cibles IP Fargate. Lorsque vous utilisez des plug-ins CNI compatibles avec Alternate, le contrôleur peut uniquement équilibrer la charge par rapport aux cibles des instances, sauf si vous équilibrez la charge par rapport aux nœuds hybrides Amazon EKS. Pour les nœuds hybrides, le contrôleur peut équilibrer la charge des cibles IP. Pour plus d'informations sur les types de cibles de Network Load Balancer, veuillez consulter la section Type de cible dans le guide de l'utilisateur de Network Load Balancer
- Si vous souhaitez ajouter des balises à l'équilibreur de charge lors de sa création ou après sa création, ajoutez l'annotation suivante dans votre spécification de service. Pour plus d'informations, consultez AWS Resource Tags dans la documentation du AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- Vous pouvez affecter des adresses IP Elastic au Network Load Balancer en ajoutant l'annotation suivante. Remplacez les *`example values`* par les `Allocation IDs` de vos adresses IP

élastiques. Le nombre de `Allocation IDs` doit correspondre au nombre de sous-réseaux utilisés pour l'équilibreur de charge. Pour plus d'informations, consultez la documentation du contrôleur AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-
xxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS ajoute une règle entrante au groupe de sécurité du nœud pour le trafic client et une règle pour chaque sous-réseau d'équilibreur de charge du VPC afin de vérifier l'état de chaque Network Load Balancer que vous créez. Le déploiement d'un service de type `LoadBalancer` peut échouer si Amazon EKS tente de créer des règles qui dépassent le quota du nombre maximal de règles autorisées pour un groupe de sécurité. Pour plus d'informations, consultez Groupes de sécurité dans Quotas de VPC Amazon dans le guide de l'utilisateur d'Amazon VPC. Tenez compte des options suivantes pour réduire les possibilités de dépassement du nombre maximum de règles pour un groupe de sécurité :
  - Demandez une augmentation dans vos règles par quota de groupe de sécurité. Pour plus d'informations, consultez Demande d'une augmentation de quota dans le Guide de l'utilisateur de Service Quotas.
  - Utilisez des cibles IP plutôt que des cibles d'instance. Avec les cibles IP, vous pouvez partager des règles pour les mêmes ports cibles. Vous pouvez spécifier manuellement les sous-réseaux des équilibreurs de charge avec une annotation. Pour plus d'informations, consultez la section Annotations sur GitHub.
  - Utilisez une entrée (ingress), au lieu d'un service de type `LoadBalancer`, pour envoyer le trafic vers votre service. L' AWS Application Load Balancer nécessite moins de règles que les Network Load Balancer. Vous pouvez partager un ALB sur plusieurs entrées. Pour de plus amples informations, veuillez consulter the section called “Répartition de la charge des applications”. Vous ne pouvez pas partager un Network Load Balancer entre plusieurs services.
  - Déployez vos clusters dans plusieurs comptes.
- Si vos pods s'exécutent sous Windows dans un cluster Amazon EKS, un seul service doté d'un équilibreur de charge peut prendre en charge jusqu'à 1 024 pods principaux. Chaque pod a sa propre adresse IP.
- Nous recommandons de créer uniquement de nouveaux Network Load Balancers à l'aide du AWS Load Balancer Controller. Toute tentative de remplacement des équilibreurs de charge réseau existants créés à l'aide du contrôleur d'équilibrage de charge du fournisseur de AWS cloud peut entraîner la création de plusieurs équilibreurs de charge réseau susceptibles de provoquer des interruptions de service des applications.

Créer un équilibreur de charge de réseau

Vous pouvez créer un équilibreur de charge de réseau avec des cibles IP ou d'instance.

Création d'un équilibreur de charge réseau — IP Targets

- Vous pouvez utiliser des cibles IP avec des pods déployés sur des EC2 nœuds Amazon, Fargate ou Amazon EKS Hybrid Nodes. Votre service Kubernetes doit être créé en tant que type `LoadBalancer`. Pour plus d'informations, consultez la section Type LoadBalancer dans la documentation de Kubernetes.

  Pour créer un équilibreur de charge qui utilise des cibles IP, ajoutez l'annotation suivante à un manifeste du service et déployez votre service. La `external` valeur de `aws-load-balancer-type` est ce qui pousse le AWS Load Balancer Controller, plutôt que le contrôleur d'équilibrage de charge du fournisseur de AWS cloud, à créer le Network Load Balancer. Vous pouvez visualiser un exemple de manifeste de service avec les annotations.

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
  ```

  > **Note**
  >
  > Si vous équilibrez la charge dans `IPv6` Pods, ajoutez l'annotation suivante. Vous ne pouvez effectuer un équilibrage de charge sur `IPv6` que vers des cibles IP, pas vers des cibles d'instance. Sans cette annotation, l'équilibrage de charge passe par `IPv4`.

  ```
  service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
  ```

  Les Network Load Balancer sont créés avec le `internal` `aws-load-balancer-scheme`, par défaut. Vous pouvez lancer des équilibreurs de charge réseau dans n'importe quel sous-réseau du VPC de votre cluster, y compris les sous-réseaux qui n'ont pas été spécifiés lors de la création de votre cluster.

  Kubernetes examine la table de routage de vos sous-réseaux afin d'identifier s'ils sont publics ou privés. Les sous-réseaux publics dispose d'un acheminement direct vers Internet en utilisant une passerelle Internet. Ce n'est pas le cas des sous-réseaux privés.

Si vous souhaitez créer un Network Load Balancer dans un sous-réseau public pour équilibrer la charge des nœuds Amazon EC2 (Fargate ne peut être que privé), spécifiez-le avec l'annotation suivante : `internet-facing`

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> Pour des raisons de rétrocompatibilité, l'annotation `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` est toujours prise en charge. Cependant, nous vous recommandons d'utiliser les annotations précédentes pour les nouveaux équilibreurs de charge au lieu de `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`.

> **Important**
>
> Ne modifiez pas les annotations après la création de votre service. Si vous devez le modifier, supprimez l'objet de service et créez-le à nouveau avec la valeur souhaitée pour cette annotation.

Créer un équilibreur de charge réseau — Instance Targets

- Le contrôleur d'équilibrage de charge du fournisseur de AWS cloud crée des équilibreurs de charge réseau avec des cibles d'instance uniquement. La version `2.2.0` et les versions ultérieures du AWS Load Balancer Controller créent également des Network Load Balancers avec des cibles d'instance. Nous vous recommandons de l'utiliser, plutôt que le contrôleur d'équilibrage de charge du fournisseur de AWS cloud, pour créer de nouveaux équilibreurs de charge réseau. Vous pouvez utiliser des cibles d'instance Network Load Balancer avec des pods déployés sur des EC2 nœuds Amazon, mais pas sur Fargate. Pour équilibrer la charge du trafic réseau entre les pods déployés sur Fargate, vous devez utiliser des cibles IP.

  Pour déployer un Network Load Balancer sur un sous-réseau privé, votre spécification de service doit comporter les annotations suivantes. Vous pouvez visualiser un exemple de manifeste de service avec les annotations. La `external` valeur de `aws-load-balancer-type` est ce qui

pousse le AWS Load Balancer Controller, plutôt que le contrôleur d'équilibrage de charge du fournisseur de AWS cloud, à créer le Network Load Balancer.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Les Network Load Balancer sont créés avec le `internal` `aws-load-balancer-scheme`, par défaut. Pour les Network Load Balancer internes, votre cluster Amazon EKS doit être configuré pour utiliser au moins un sous-réseau privé dans votre VPC. Kubernetes examine la table de routage de vos sous-réseaux afin d'identifier s'ils sont publics ou privés. Les sous-réseaux publics dispose d'un acheminement direct vers Internet en utilisant une passerelle Internet. Ce n'est pas le cas des sous-réseaux privés.

Si vous souhaitez créer un Network Load Balancer dans un sous-réseau public pour équilibrer la charge des EC2 nœuds Amazon, spécifiez-le à l'`internet-facing`aide de l'annotation suivante :

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> Ne modifiez pas les annotations après la création de votre service. Si vous devez le modifier, supprimez l'objet de service et créez-le à nouveau avec la valeur souhaitée pour cette annotation.

(Facultatif) Déployer un exemple d'application

- Au moins un sous-réseau public ou privé dans votre VPC de cluster.
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.

  1. Si vous effectuez un déploiement sur Fargate, assurez-vous qu'un sous-réseau privé est disponible dans votre VPC et créez un profil Fargate. Si vous ne déployez pas sur Fargate, ignorez cette étape. Vous pouvez créer le profil en exécutant la commande suivante ou dans la AWS Management Console en utilisant les mêmes valeurs pour `name` et `namespace` que dans la commande. Remplacez les *`example values`* par vos propres valeurs.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name nlb-sample-app \
    --namespace nlb-sample-app
```

2. Déployez un exemple d'application.

   a. Créez un espace de nom pour l'application.

   ```
   kubectl create namespace nlb-sample-app
   ```

   b. Enregistrez le contenu suivant dans un fichier nommé `sample-deployment.yaml` sur votre ordinateur.

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nlb-sample-app
     namespace: nlb-sample-app
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         labels:
           app: nginx
       spec:
         containers:
           - name: nginx
             image: public.ecr.aws/nginx/nginx:1.23
             ports:
               - name: tcp
                 containerPort: 80
   ```

   c. Appliquez le fichier manifeste à votre cluster.

   ```
   kubectl apply -f sample-deployment.yaml
   ```

3. Créez un service avec un Network Load Balancer interne qui équilibre la charge vers les cibles IP.

   a. Enregistrez le contenu suivant dans un `sample-service.yaml` fichier nommé file sur votre ordinateur. Si vous effectuez un déploiement sur des nœuds Fargate, supprimez la ligne. `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing`

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

   b. Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f sample-service.yaml
```

4. Vérifiez que le service a été déployé.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

L'exemple qui suit illustre un résultat.

```
NAME            TYPE           CLUSTER-IP         EXTERNAL-IP
                                                PORT(S)        AGE
sample-service  LoadBalancer   10.100.240.137   k8s-nlbsampl-nlbsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com  80:32400/TCP   16h
```

> **Note**
>
> Les valeurs pour *10.100.240.137* et *xxxxxxxxxx* - *xxxxxxxxxxxxxxxx* seront différentes de celles de l'exemple de sortie (elles seront propres à votre équilibreur de charge) et *us-west-2* peuvent être différentes pour vous, selon AWS la région dans laquelle se trouve votre cluster.

5. Ouvrez l'Amazon EC2 AWS Management Console. Sélectionnez Target Groups (Groupes cibles) (sous Load Balancing [Équilibrage de charge]) dans le panneau de navigation de gauche. Dans la colonne Nom, sélectionnez le nom du groupe cible dont la valeur de la colonne de l'équilibreur de charge correspond à une partie du nom de la `EXTERNAL-IP` colonne de sortie de l'étape précédente. Par exemple, vous devez sélectionner le groupe cible nommé `k8s-default-samplese-`*xxxxxxxxxx* si votre sortie est identique à la sortie précédente. Le Type de cible est `IP`, car cela a été spécifié dans l'exemple de manifeste de service.
6. Sélectionnez votre Target group (Groupe cible), puis cliquez sur l'onglet Targets (Cibles). Sous Cibles enregistrées, vous devez voir trois adresses IP des trois réplicas déployés à une étape précédente. Attendez que le statut de toutes les cibles soit sain avant de continuer. L'affichage du statut `healthy` pour toutes les cibles peut prendre plusieurs minutes. Les cibles peuvent avoir l'état `unhealthy` avant qu'il soit remplacé par `healthy`.
7. Envoyer le trafic vers le service en remplaçant *xxxxxxxxxx-xxxxxxxxxxxxxxxx* et *us-west-2* par les valeurs renvoyées dans la sortie de l'étape précédente pour `EXTERNAL-IP`. Si vous avez effectué le déploiement sur un sous-réseau privé, vous devez afficher la page depuis un appareil au sein de votre VPC, tel qu'un hôte bastion. Pour plus d'informations, consultez la section Hôtes bastions Linux sur AWS.

   ```
   curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
   ```

   L'exemple qui suit illustre un résultat.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

8. Lorsque vous avez terminé avec les exemples de déploiement, de service et d'espace de noms, supprimez-les.

```
kubectl delete namespace nlb-sample-app
```

Acheminez le trafic d'applications et le trafic HTTP avec des équilibreurs de charge d'application

> **Note**
>
> Nouveau : le mode automatique d'Amazon EKS automatise les tâches de routine pour l'équilibrage de charge. Pour plus d'informations, consultez :
>
> - the section called “Déployer un équilibreur de charge”
> - the section called “Créer une classe d'entrée”

Lorsque vous créez un Kubernetes`ingress`, un Application AWS Load Balancer (ALB) est configuré pour équilibrer la charge du trafic des applications. Pour en savoir plus, consultez Qu'est-ce qu'un Application Load Balancer ? dans le guide de l'utilisateur des équilibreurs de charge d'application et Ingress dans la documentation de Kubernetes. ALBs peut être utilisé avec des pods déployés sur des nœuds ou sur AWS Fargate. Vous pouvez déployer un ALB sur des sous-réseaux publics ou privés.

Le trafic des applications est équilibré au `L7` du modèle OSI. Pour équilibrer la charge du trafic réseau au `L4`, déployez un `service` Kubernetes de type `LoadBalancer`. Ce type fournit un AWS Network Load Balancer. Pour de plus amples informations, veuillez consulter the section called “Répartition de charge réseau”. Pour en savoir plus sur les différences entre les deux types d'équilibrage de charge, consultez les fonctionnalités d'Elastic Load Balancing sur le AWS site Web.

Prérequis

Pour pouvoir équilibrer la charge du trafic d'une application, vous devez remplir les conditions suivantes.

- Disposer d'un cluster. Si vous n'avez pas de cluster existant, consultezMise en route. Si vous devez mettre à jour la version d'un cluster existant, consultez the section called “Mettre à jour la version de Kubernetes”.

- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.
- Au moins deux sous-réseaux dans des zones de disponibilité différentes. Le AWS Load Balancer Controller choisit un sous-réseau dans chaque zone de disponibilité. Lorsque plusieurs sous-réseaux étiquetés sont trouvés dans une zone de disponibilité, le contrôleur choisit le sous-réseau dont l'ID vient en premier par ordre lexicographique. Chaque sous-réseau doit disposer au moins huit adresses IP disponibles.

  Si vous utilisez plusieurs groupes de sécurité attachés au nœud de travail, un seul groupe de sécurité doit être balisé comme suit. Remplacez *my-cluster* par le nom de votre cluster.

  - Clé : `kubernetes.io/cluster/<my-cluster>`
  - Valeur : `shared` ou `owned`
- Si vous utilisez la version du AWS Load Balancer Controller `2.1.1` ou une version antérieure, les sous-réseaux doivent être balisés au format suivant. Si vous utilisez une version `2.1.2` ou une version ultérieure, le balisage est facultatif. Cependant, nous vous recommandons d'étiqueter un sous-réseau dans les cas suivants. Plusieurs clusters s'exécutent dans le même VPC ou plusieurs AWS services partagent des sous-réseaux au sein d'un VPC. Sinon, vous souhaitez avoir plus de contrôle sur l'endroit où les équilibreurs de charge sont alloués pour chaque cluster. Remplacez *my-cluster* par le nom de votre cluster.

  - Clé : `kubernetes.io/cluster/<my-cluster>`
  - Valeur : `shared` ou `owned`
- Vos sous-réseaux publics et privés doivent répondre aux critères suivants : C'est le cas sauf si vous spécifiez explicitement un sous-réseau IDs sous forme d'annotation sur un service ou un objet d'entrée. Supposons que vous provisionnez des équilibreurs de charge en spécifiant explicitement le sous-réseau IDs sous forme d'annotation sur un service ou un objet d'entrée. Dans ce cas, Kubernetes et le contrôleur d'équilibrage de AWS charge utilisent directement ces sous-réseaux pour créer l'équilibreur de charge et les balises suivantes ne sont pas requises.

  - Sous-réseaux privés : doivent être étiquetés dans le format suivant. Cela permet à Kubernetes et au contrôleur d'équilibreur de AWS charge de savoir que les sous-réseaux peuvent être utilisés pour les équilibreurs de charge internes. Si vous utilisez `eksctl` un AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de AWS CloudFormation VPC Amazon EKS, consultez. the section called "Création d'un VPC"

    - Clé : `kubernetes.io/role/internal-elb`

  - Valeur : 1
- Sous-réseau publics : doivent être étiquetés dans le format suivant. Ainsi, Kubernetes sait qu'il doit utiliser uniquement les sous-réseaux qui ont été spécifiés pour les équilibreurs de charge externes. De cette façon, Kubernetes ne choisit pas de sous-réseau public dans chaque zone de disponibilité (lexicographiquement en fonction de leur ID de sous-réseau). Si vous utilisez `eksctl` un AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de AWS CloudFormation VPC Amazon EKS, consultez. the section called "Création d'un VPC"
  - Clé : `kubernetes.io/role/elb`
  - Valeur : 1

Si les balises de rôle de sous-réseau ne sont pas explicitement ajoutées, le contrôleur de service Kubernetes examine la table de routage des sous-réseaux VPC de votre cluster. Cela permet de déterminer si le sous-réseau est privé ou public. Nous vous recommandons de ne pas vous fier à ce comportement. Ajoutez plutôt explicitement les identifications de rôle privées ou publiques. Le AWS Load Balancer Controller n'examine pas les tables de routage. Il est également nécessaire que les identifications privées et publiques soient présentes pour que la découverte automatique fonctionne.

- Le AWS Load Balancer Controller crée ALBs les AWS ressources de support nécessaires chaque fois qu'une ressource d'entrée Kubernetes est créée sur le cluster avec l'annotation. `kubernetes.io/ingress.class: alb` La ressource d'entrée configure l'ALB pour acheminer le trafic HTTP ou HTTPS vers différents pods au sein du cluster. Pour vous assurer que vos objets d'entrée utilisent le AWS Load Balancer Controller, ajoutez l'annotation suivante à votre spécification d'entrée Kubernetes. Pour plus d'informations, consultez Spécification d'entrée sur GitHub.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> Si vous équilibrez la charge avec `IPv6` Pods, ajoutez l'annotation suivante à vos spécifications d'entrée. Vous ne pouvez effectuer un équilibrage de charge sur `IPv6` que

vers des cibles IP, pas vers des cibles d'instance. Sans cette annotation, l'équilibrage de charge passe par `IPv4`.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- Le AWS Load Balancer Controller prend en charge les modes de trafic suivants :
  - Instance : enregistre les nœuds de votre cluster comme cibles de l'ALB. Le trafic atteignant l'ALB est acheminé vers `NodePort` votre service, puis transmis par proxy vers vos Pods. Il s'agit du mode de trafic par défaut. Vous pouvez également le spécifier explicitement avec l'annotation `alb.ingress.kubernetes.io/target-type: instance`.

    > **Note**
    > Votre service Kubernetes doit spécifier le `LoadBalancer` type `NodePort` ou pour utiliser ce mode de trafic.

  - IP : enregistre les pods comme cibles pour l'ALB. Le trafic atteignant l'ALB est directement acheminé vers Pods pour votre service. Vous devez spécifier l'annotation `alb.ingress.kubernetes.io/target-type: ip` pour pouvoir utiliser ce mode de trafic. Le type de cible IP est requis lorsque les pods cibles s'exécutent sur des nœuds hybrides Fargate ou Amazon EKS.
- Pour étiqueter une balise ALBs créée par le contrôleur, ajoutez l'annotation suivante au contrôleur :`alb.ingress.kubernetes.io/tags`. Pour une liste de toutes les annotations disponibles prises en charge par le AWS Load Balancer Controller, voir Annotations d'entrée sur. GitHub
- La mise à niveau ou la rétrogradation de la version du contrôleur ALB peut introduire des changements de rupture pour les fonctionnalités qui en dépendent. Pour plus d'informations sur les changements importants introduits dans chaque version, consultez les notes de mise à jour du contrôleur ALB sur GitHub.

Réutilisation ALBs avec des groupes d'entrée

Vous pouvez partager un équilibreur de charge d'application entre plusieurs ressources de service à l'aide `IngressGroups` de.

Pour joindre une ressource d'entrée à un groupe, ajoutez l'annotation suivante à une spécification de ressource d'entrée Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

Le nom du groupe doit :

- Contenir 63 caractères maximum.
- Contenir des minuscules, des chiffres, des `-` et des `.`.
- Commencer et se terminer par un chiffre ou une lettre.

Le contrôleur fusionne automatiquement les règles d'entrée pour toutes les entrées du même groupe d'entrées. Il les prend en charge avec un seul ALB. La plupart des annotations qui sont définies sur une ressource d'entrée ne s'appliquent qu'aux chemins définis par cette ressource. Par défaut, les ressources d'entrée n'appartiennent à aucun groupe d'entrée.

> **Warning**
>
> Risque de sécurité potentiel
>
> Spécifiez un groupe d'entrée pour une entrée uniquement lorsque tous les utilisateurs de Kubernetes autorisés à créer ou à modifier des ressources d'entrée se situent dans la même limite de confiance. Si vous ajoutez l'annotation avec un nom de groupe, d'autres utilisateurs Kubernetes peuvent créer ou modifier leurs ressources d'entrée pour appartenir au même groupe d'entrée. Cela peut entraîner un comportement indésirable, comme l'écrasement des règles existantes par des règles de priorité supérieure.

Vous pouvez ajouter le numéro d'ordre de votre ressource d'entrée.

```
alb.ingress.kubernetes.io/group.order: '10'
```

Le numéro peut être 1-1000. Le numéro le plus bas de toutes les ressources d'entrée d'un même groupe d'entrée est évalué en premier. Toutes les ressources d'entrée sans cette annotation sont évaluées avec la valeur zéro. Les règles dupliquées avec un numéro supérieur peuvent écraser les règles avec un numéro inférieur. Par défaut, l'ordre des règles entre les ressources d'entrée d'un même groupe d'entrée est déterminé de manière lexicographique sur la base de l'espace de noms et du nom.

> **Important**
>
> Assurez-vous que chaque ressource d'un groupe d'entrée dispose d'un numéro de priorité unique. Vous ne pouvez pas avoir de numéros de commande dupliqués entre les entrées.

(Facultatif) Déployer un exemple d'application

- Au moins un sous-réseau public ou privé dans votre VPC de cluster.
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.

Vous pouvez exécuter l'exemple d'application sur un cluster doté de EC2 nœuds Amazon, de Fargate Pods ou des deux.

1. Si vous ne déployez pas sur Fargate, ignorez cette étape. Si vous effectuez un déploiement sur Fargate, créez un profil Fargate. Vous pouvez créer le profil en exécutant la commande suivante ou dans la AWS Management Console en utilisant les mêmes valeurs pour `name` et `namespace` que dans la commande. Remplacez les *`example values`* par vos propres valeurs.

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name alb-sample-app \
       --namespace game-2048
   ```

2. Déployez le jeu 2048 comme exemple d'application pour vérifier que le AWS Load Balancer Controller crée AWS un ALB à la suite de l'objet d'entrée. Suivez les étapes correspondant au type de sous-réseau vers lequel vous effectuez le déploiement.

   a. Si vous effectuez un déploiement sur des Pods dans un cluster que vous avez créé avec la `IPv6` famille, passez à l'étape suivante.

      - Publique :

      ```
      kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-
      balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
      ```

- Privé :

  A. Téléchargez le manifeste.

  ```
  curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
  ```

  B. Modifiez le fichier et trouvez la ligne qui contient `alb.ingress.kubernetes.io/scheme: internet-facing`.

  C. Remplacez *`internet-facing`* par `internal` et enregistrez le fichier.

  D. Appliquez le manifeste à votre cluster.

  ```
  kubectl apply -f 2048_full.yaml
  ```

b. Si vous effectuez un déploiement sur des Pods dans un cluster que vous avez créé avec la IPv6 famille, procédez comme suit.

   i. Téléchargez le manifeste.

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
   ```

   ii. Ouvrez le fichier dans un éditeur et ajoutez la ligne suivante aux annotations de la spécification d'entrée.

   ```
   alb.ingress.kubernetes.io/ip-address-type: dualstack
   ```

   iii. Si vous équilibrez la charge sur des pods internes, plutôt que sur des pods connectés `alb.ingress.kubernetes.io/scheme:` *`internet-facing`* à Internet, modifiez la ligne indiquant `alb.ingress.kubernetes.io/scheme: internal`

   iv. Enregistrez le fichier.

   v. Appliquez le manifeste à votre cluster.

   ```
   kubectl apply -f 2048_full.yaml
   ```

3. Après quelques minutes, vérifiez que la ressource d'entrée a été créée en utilisant la commande suivante.

```
kubectl get ingress/ingress-2048 -n game-2048
```

L'exemple qui suit illustre un résultat.

```
NAME           CLASS    HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> Si vous avez créé l'équilibreur de charge dans un sous-réseau privé, la valeur sous `ADDRESS` dans la sortie précédente est préfacée avec `internal-`.

Si votre entrée n'a pas été créée correctement après plusieurs minutes, exécutez la commande suivante pour afficher les journaux du AWS Load Balancer Controller. Ces journaux peuvent contenir des messages d'erreur que vous pouvez utiliser pour diagnostiquer les problèmes de votre déploiement.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-
controller
```

1. Si vous l'avez déployé dans un sous-réseau public, ouvrez un navigateur et naviguez vers l'URL `ADDRESS` de la sortie de commande précédente pour voir l'exemple d'application. Si rien ne s'affiche, actualisez votre navigateur et réessayez. Si vous avez effectué le déploiement sur un sous-réseau privé, vous devez afficher la page depuis un appareil de votre VPC, tel qu'un hôte bastion. Pour plus d'informations, consultez la section Hôtes bastions Linux sur AWS.

2. Lorsque vous avez terminé de tester l'exemple d'application, supprimez-le avec les commandes suivantes.
   - Si vous avez appliqué le manifeste plutôt que d'appliquer une copie que vous avez téléchargée, utilisez la commande suivante.

     ```
     kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.3/docs/examples/2048/2048_full.yaml
     ```

   - Si vous avez téléchargé et modifié le manifeste, utilisez la commande suivante.

     ```
     kubectl delete -f 2048_full.yaml
     ```

Restreindre les adresses IP externes pouvant être attribuées aux services

Les services Kubernetes peuvent être atteints depuis l'intérieur d'un cluster via :

- Une adresse IP de cluster affectée automatiquement par Kubernetes
- Toute adresse IP que vous spécifiez pour la propriété `externalIPs` dans une spécification de service. Les adresses IP externes ne sont pas gérées par Kubernetes et relèvent de la responsabilité de l'administrateur du cluster. Les adresses IP externes spécifiées avec `externalIPs` sont différentes de l'adresse IP externe affectée à un service de type `LoadBalancer` par un fournisseur de cloud.

Pour en savoir plus sur les services Kubernetes, consultez Service dans la documentation Kubernetes. Vous pouvez restreindre les adresses IP qui peuvent être spécifiées pour `externalIPs` dans une spécification de service.

1. Déployez `cert-manager` pour gérer les certificats webhook. Pour plus d'informations, consultez la documentation cert-manager.

   ```
   kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
   ```

2. Vérifiez que les `cert-manager` pods fonctionnent.

```
kubectl get pods -n cert-manager
```

L'exemple qui suit illustre un résultat.

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
```

3. Passez en revue vos services existants pour vous assurer qu'aucun d'entre eux ne possède d'adresse IP externe qui ne soit contenue dans le bloc CIDR auquel vous souhaitez limiter les adresses.

```
kubectl get services -A
```

L'exemple qui suit illustre un résultat.

```
NAMESPACE                      NAME                                      TYPE
 CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
cert-manager                   cert-manager                              ClusterIP
 10.100.102.137   <none>          9402/TCP        20m
cert-manager                   cert-manager-webhook                      ClusterIP
 10.100.6.136     <none>          443/TCP         20m
default                        kubernetes                                ClusterIP
 10.100.0.1       <none>          443/TCP         2d1h
externalip-validation-system   externalip-validation-webhook-service     ClusterIP
 10.100.234.179   <none>          443/TCP         16s
kube-system                    kube-dns                                  ClusterIP
 10.100.0.10      <none>          53/UDP,53/TCP   2d1h
my-namespace                   my-service                                ClusterIP
 10.100.128.10    192.168.1.1     80/TCP          149m
```

Si l'une des valeurs correspond à des adresses IP qui ne se trouvent pas dans le bloc auquel vous souhaitez restreindre l'accès, vous devez modifier les adresses pour qu'elles se trouvent dans le bloc et redéployer les services. Par exemple, le `my-service` service indiqué dans la sortie précédente possède une adresse IP externe qui ne figure pas dans l'exemple de bloc CIDR de l'étape 5.

4. Téléchargez le manifeste du webhook IP externe. Vous pouvez également afficher le code source pour le webhook sur GitHub.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. Spécifiez des blocs d'adresse CIDR. Ouvrez le fichier téléchargé dans l'éditeur et supprimez le \# au début des lignes suivantes.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

   Remplacez `10.0.0.0/8` par votre propre bloc d'adresse CIDR. Vous pouvez spécifier autant de blocs que vous le souhaitez. Si vous spécifiez plusieurs blocs, ajoutez une virgule entre les blocs.

6. Si votre cluster ne se trouve pas dans la `us-west-2` AWS régionus-west-2, remplacez`602401143452`, et `amazonaws.com` dans le fichier par les commandes suivantes. Avant d'exécuter les commandes, remplacez *`region-code`* et *`111122223333`* par la valeur de votre AWS région dans la liste dans Afficher les registres d'images de conteneurs Amazon pour les modules complémentaires Amazon EKS.

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. Appliquez le manifeste à votre cluster.

```
kubectl apply -f externalip-webhook.yaml
```

   Toute tentative de déploiement d'un service sur votre cluster avec une adresse IP spécifiée ne figurant pas dans les blocs que vous avez spécifiés à l'étape Spécifier les blocs CIDR échouera. `externalIPs`

Copier une image de conteneur d'un référentiel vers un autre référentiel

Cette rubrique explique comment extraire une image de conteneur d'un référentiel auquel vos nœuds n'ont pas accès et comment transférer l'image vers un référentiel auquel vos nœuds ont accès. Vous pouvez transmettre l'image à Amazon ECR ou à un référentiel alternatif auquel vos nœuds ont accès.

- Docker Engine installé et configuré sur votre ordinateur. Pour connaître la marche à suivre, consultez la rubrique Installer Docker Engine de la documentation Docker.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un point de terminaison VPC d'interface pour Amazon ECR si vous souhaitez que vos nœuds extraient des images de conteneurs ou les envoient vers un référentiel Amazon ECR privé via le réseau d'Amazon. Pour de plus amples informations, veuillez consulter Créer des points de terminaison de VPC pour Amazon ECR dans le Guide de l'utilisateur Amazon Elastic Container Registry.

Effectuez les étapes suivantes pour extraire une image de conteneur d'un référentiel et la transférer vers votre propre référentiel. Dans les exemples suivants fournis dans cette rubrique, l'image du plug-in Amazon VPC CNI pour Kubernetes Metrics Helper est extraite. Lorsque vous suivez ces étapes, assurez-vous de remplacer les *`example values`* par vos propres valeurs.

1. Si vous ne disposez pas encore d'un référentiel Amazon ECR ou d'un autre référentiel, créez-en un auquel vos nœuds ont accès. La commande suivante crée un référentiel privé Amazon ECR. Un nom de référentiel privé Amazon ECR doit commencer par une lettre. Il peut uniquement contenir des lettres minuscules, des chiffres, des tirets (-), des traits de soulignement (_) et des barres obliques (/). Pour de plus amples informations, veuillez consulter Créer un référentiel privé dans le Guide de l'utilisateur Amazon Elastic Container Registry.

   Vous pouvez remplacer.*`cni-metrics-helper`* par ce que vous souhaitez. Il est recommandé de créer un référentiel distinct pour chaque image. Nous vous le recommandons, car les identifications d'image doivent être uniques au sein d'un référentiel. Remplacez *`region-code`* par une AWS région prise en charge par Amazon ECR.

   ```
   aws ecr create-repository --region region-code --repository-name cni-metrics-helper
   ```

2. Déterminez le registre, le référentiel et l'identification (facultatif) de l'image que vos nœuds doivent extraire. Ces informations se trouvent dans le format `registry/repository[:tag]`.

   De nombreuses rubriques Amazon EKS sur l'installation d'images nécessitent que vous appliquiez un fichier manifeste ou que vous installiez l'image à l'aide des Charts de Helm. Toutefois, avant d'appliquer un fichier manifeste ou d'installer un graphique Helm, consultez d'abord le contenu du manifeste ou du `values.yaml` fichier du graphique. De cette façon, vous pouvez déterminer le registre, le référentiel et l'identification à extraire.

   Par exemple, vous pouvez trouver la ligne suivante dans le fichier manifeste du plug-in Amazon VPC CNI pour Kubernetes Metrics Helper. Le registre est `602401143452.dkr.ecr.us-west-2.amazonaws.com`, qui est un registre privé Amazon ECR. Le référentiel est `cni-metrics-helper`.

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   Vous pouvez voir les variantes suivantes pour un emplacement d'image :

   - Seul `repository-name:tag`. Dans ce cas, `docker.io` est généralement le registre, mais non spécifié puisque Kubernetes l'ajoute par défaut à un nom de référentiel si aucun registre n'est spécifié.
   - `repository-name/repository-namespace/repository:tag`. Un espace de noms de référentiel est facultatif, mais est parfois spécifié par le propriétaire du référentiel pour catégoriser les images. Par exemple, toutes les EC2 images Amazon de la galerie publique Amazon ECR utilisent l'espace de `aws-ec2` noms.

     Avant d'installer une image avec Helm, affichez le fichier `values.yaml` Helm pour déterminer l'emplacement de l'image. Par exemple, le fichier values.yaml du plugin Amazon VPC CNI pour Kubernetes Metrics Helper inclut les lignes suivantes.

     ```
     image:
       region: us-west-2
       tag: v1.12.6
       account: "602401143452"
       domain: "amazonaws.com"
     ```

3. Extrayez l'image conteneur spécifiée dans le fichier manifeste.

   a. Si vous utilisez un registre public, tel que l'Amazon ECR Public Gallery, vous pouvez passer à la sous-étape suivante, car l'authentification n'est pas requise. Dans cet exemple, vous

vous authentifiez auprès d'un registre privé Amazon ECR contenant le référentiel pour l'image d'assistance des métriques CNI. Amazon EKS conserve l'image dans chaque registre répertorié dans Afficher les registres d'images de conteneurs Amazon pour les modules complémentaires Amazon EKS. Vous pouvez vous authentifier auprès de n'importe quel registre en remplaçant *602401143452* et *region-code* par les informations d'un registre différent. Il existe un registre distinct pour chaque AWS région dans laquelle Amazon EKS est pris en charge.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
```

b. Extrayez l'image. Dans cet exemple, vous effectuez une extraction à partir du registre auquel vous vous êtes authentifié au cours de la sous-étape précédente. Remplacez *602401143452* et *region-code* par les informations que vous avez fournies à la sous-étape précédente.

```
docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

4. Étiquetez l'image que vous avez extraite avec votre registre, votre référentiel et votre identification. L'exemple suivant suppose que vous avez extrait l'image du fichier manifeste et que vous allez la transmettre au référentiel privé Amazon ECR que vous avez créé à la première étape. Remplacez *111122223333* par votre ID de compte. *region-code*Remplacez-le par la AWS région dans laquelle vous avez créé votre référentiel privé Amazon ECR.

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/
cni-metrics-helper:v1.12.6
```

5. Authentifiez auprès de votre registre. Dans cet exemple, vous vous authentifiez auprès du registre privé Amazon ECR que vous avez créé à la première étape. Pour plus d'informations, veuillez consulter Authentification de registre dans le Guide de l'utilisateur Amazon Elastic Container Registry.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. Transmettez l'image à votre référentiel. Dans cet exemple, vous transmettez l'image au référentiel privé Amazon ECR que vous avez créé à la première étape. Pour plus d'informations, consultez Transmission d'une image Dockerdans le Guide de l'utilisateur Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. Mettez à jour le fichier manifeste que vous avez utilisé pour déterminer l'image lors d'une étape précédente avec le `registry/repository:tag` pour l'image que vous avez transmise. Si vous effectuez l'installation à l'aide d'un graphique Helm, il est souvent possible de spécifier le`registry/repository:tag`. Lors de l'installation du graphique, spécifiez le `registry/repository:tag` pour l'image que vous avez transmise à votre référentiel.

Afficher les registres d'images de conteneurs Amazon pour les modules complémentaires Amazon EKS

Lorsque vous déployez des modules complémentaires AWS Amazon EKS dans votre cluster, vos nœuds extraient les images de conteneur requises à partir du registre spécifié dans le mécanisme d'installation du module complémentaire, comme un manifeste d'installation ou un fichier Helm `values.yaml`. Les images sont extraites à partir d'un référentiel privé Amazon EKS Amazon ECR. Amazon EKS réplique les images dans un référentiel dans chaque AWS région prise en charge par Amazon EKS. Vos nœuds peuvent extraire l'image du conteneur sur Internet à partir de l'un des registres suivants. Vos nœuds peuvent également extraire l'image sur le réseau d'Amazon si vous avez créé un point de terminaison VPC d'interface pour Amazon ECR (AWS PrivateLink) dans votre VPC. Les registres nécessitent une authentification avec un compte AWS IAM. Vos nœuds s'authentifient à l'aide du rôle IAM du nœud Amazon EKS, qui dispose des autorisations associées dans la politique IAM EC2 ContainerRegistryReadOnly gérée par Amazon.

AWS Région	Registre
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-est-2	533267051163.dkr. ecr.ap-east-1.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com

AWS Région	Registre
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-sud-est 7	121268973566.dkr. ecr.ap-southeast-7.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn/
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn/
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com

AWS Région	Registre
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
mx-central-1	730335286997.dkr. ecr.mx-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com

AWS Région	Registre
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com
us-gov-east-1	151742754352.dkr.ecr. us-gov-east-1. amazonaws.com
us-gov-west-1	013241004608.dkr.ecr. us-gov-west-1. amazonaws.com

Modules complémentaires Amazon EKS

Un module complémentaire est un logiciel qui fournit des capacités opérationnelles de prise en charge aux applications Kubernetes, mais qui n'est pas spécifique à l'application. Cela inclut des logiciels tels que des agents d'observabilité ou des pilotes Kubernetes qui permettent au cluster d'interagir avec les AWS ressources sous-jacentes pour la mise en réseau, le calcul et le stockage. Les logiciels complémentaires sont généralement conçus et maintenus par la communauté Kubernetes, par des fournisseurs de cloud tels que ou par des fournisseurs AWS tiers. Amazon EKS installe automatiquement des modules complémentaires autogérés tels que le plug-in Amazon VPC CNI pour Kubernetes et CoreDNS pour chaque cluster. `kube-proxy` Notez que le module complémentaire VPC CNI n'est pas compatible avec les nœuds hybrides Amazon EKS et ne se déploie pas sur les nœuds hybrides. Vous pouvez modifier la configuration par défaut des modules complémentaires et les mettre à jour quand vous le souhaitez.

Les modules complémentaires Amazon EKS fournissent l'installation et la gestion d'un ensemble organisé de modules complémentaires pour les clusters Amazon EKS. Tous les modules complémentaires Amazon EKS incluent les derniers correctifs de sécurité et corrections de bogues et sont validés AWS pour fonctionner avec Amazon EKS. Les modules complémentaires Amazon EKS vous permettent de garantir systématiquement que vos clusters Amazon EKS sont sécurisés et stables. Ils vous permettent aussi de réduire le travail nécessaire pour installer, configurer et mettre

à jour les modules complémentaires. Si un module complémentaire autogéré tel que `kube-proxy` est déjà en cours d'exécution sur votre cluster et est disponible en tant que module complémentaire Amazon EKS, vous pouvez installer le module complémentaire Amazon EKS `kube-proxy` pour commencer à bénéficier des fonctionnalités des modules complémentaires Amazon EKS.

Vous pouvez mettre à jour des champs de configuration spécifiques gérés par Amazon EKS pour les modules complémentaires Amazon EKS via l'API Amazon EKS. Une fois le module complémentaire démarré, vous pouvez également modifier les champs de configuration non gérés par Amazon EKS directement dans le cluster Kubernetes. Cela inclut la définition de champs de configuration spécifiques pour un module complémentaire, le cas échéant. Ces modifications ne sont pas remplacées par Amazon EKS une fois qu'elles sont effectuées. Ceci est rendu possible par la fonction apply côté serveur de Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Champs que vous pouvez personnaliser".

Vous pouvez utiliser des modules complémentaires Amazon EKS avec n'importe quel type de nœud Amazon EKS. Pour de plus amples informations, veuillez consulter Gérer le calcul.

Vous pouvez ajouter, mettre à jour ou supprimer des modules complémentaires Amazon EKS à l'aide de l'API Amazon EKS AWS Management Console, de la AWS CLI et`eksctl`. Vous pouvez également créer des modules complémentaires Amazon EKS à l'aide de AWS CloudFormation.

Considérations

Tenez compte des points suivants lorsque vous utilisez les modules complémentaires Amazon EKS :

- Pour configurer des modules complémentaires pour le cluster, votre principal IAM doit disposer des autorisations IAM pour utiliser des modules complémentaires. Pour plus d'informations, consultez les actions ayant `Addon` dans leur nom dans Actions définies par Amazon Elastic Kubernetes Service.
- Les modules complémentaires Amazon EKS s'exécutent sur les nœuds que vous approvisionnez ou configurez pour votre cluster. Les types de nœuds incluent EC2 les instances Amazon, Fargate et les nœuds hybrides.
- Vous pouvez modifier les champs qui ne sont pas gérés par Amazon EKS pour personnaliser l'installation d'un module complémentaire Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Champs que vous pouvez personnaliser".
- Si vous créez un cluster avec les modules AWS Management Console complémentaires Amazon EKS`kube-proxy`, Amazon VPC CNI pour Kubernetes et CoreDNS Amazon EKS sont

automatiquement ajoutés à votre cluster. Si vous utilisez `eksctl` pour créer votre cluster avec un fichier `config`, `eksctl` peut également créer le cluster avec les modules complémentaires Amazon EKS. Si vous créez votre cluster `eksctl` sans `config` fichier ou avec un autre outil, le plug-in Amazon VPC CNI autogéré `kube-proxy` pour Kubernetes et les modules complémentaires CoreDNS sont installés, plutôt que les modules complémentaires Amazon EKS. Vous pouvez les gérer vous-même ou ajouter manuellement les modules complémentaires Amazon EKS après la création du cluster. Quelle que soit la méthode que vous utilisez pour créer votre cluster, le module complémentaire VPC CNI ne s'installe pas sur les nœuds hybrides.

- Le `eks:addon-cluster-admin` `ClusterRoleBinding` lie `cluster-admin` `ClusterRole` à l'identité `eks:addon-manager` Kubernetes. Le rôle dispose des autorisations nécessaires pour l'`eks:addon-manager`identité afin de créer des espaces de noms Kubernetes et d'installer des modules complémentaires dans des espaces de noms. Si la ressource `ClusterRoleBinding` `eks:addon-cluster-admin` est supprimée, le cluster Amazon EKS continuera de fonctionner, mais Amazon EKS ne sera plus en mesure de gérer les modules complémentaires. Tous les clusters qui démarrent avec les versions de plateforme suivantes utilisent la nouvelle ressource `ClusterRoleBinding`.
- La compatibilité d'un sous-ensemble de AWS modules complémentaires EKS avec Amazon EKS Hybrid Nodes a été validée. Pour plus d'informations, consultez le tableau de compatibilité surthe section called " AWS modules complémentaires".

Considérations relatives au mode automatique d'Amazon EKS

Le mode Amazon EKS Auto inclut des fonctionnalités qui fournissent des fonctionnalités de cluster essentielles, notamment :

- Mise en réseau de pods
- Réseau de services
- DNS du cluster
- Autoscaling
- Stockage en mode bloc
- Contrôleur d'équilibrage de charge
- Agent Pod Identity
- Agent de surveillance des nœuds

Avec le calcul en mode automatique, de nombreux modules complémentaires EKS couramment utilisés deviennent redondants, tels que :

- CNI Amazon VPC
- kube-proxy
- CoreDNS
- Pilote Amazon EBS CSI
- Agent d'identité du pod EKS

Toutefois, si votre cluster combine le mode automatique avec d'autres options de calcul telles que les EC2 instances autogérées, les groupes de nœuds gérés ou AWS Fargate, ces modules complémentaires restent nécessaires. AWS a amélioré les modules complémentaires EKS avec des règles anti-affinité qui garantissent automatiquement que les modules complémentaires sont planifiés uniquement sur les types de calcul pris en charge. En outre, les utilisateurs peuvent désormais utiliser l'`DescribeAddonVersions`API des modules complémentaires EKS pour vérifier les types de calcul pris en charge pour chaque module complémentaire et ses versions spécifiques. De plus, avec le mode EKS Auto, les contrôleurs répertoriés ci-dessus fonctionnent sur AWS une infrastructure propre. Il est donc possible que vous ne les voyiez même pas dans vos comptes, sauf si vous utilisez le mode automatique EKS avec d'autres types de calcul, auquel cas vous verrez les contrôleurs que vous avez installés sur votre cluster.

Si vous envisagez d'activer le mode automatique EKS sur un cluster existant, vous devrez peut-être mettre à niveau la version de certains modules complémentaires. Pour plus d'informations, reportez-vous à the section called "Versions complémentaires requises" la section relative au mode automatique EKS.

Support

AWS publie plusieurs types de modules complémentaires avec différents niveaux de support.

- AWS Modules complémentaires : Ces modules complémentaires sont conçus et entièrement pris en charge par AWS.
  - Utilisez un AWS module complémentaire pour travailler avec d'autres AWS services, tels qu'Amazon EFS.
  - Pour de plus amples informations, veuillez consulter the section called " AWS modules complémentaires".

- AWS Modules complémentaires Marketplace : ces modules complémentaires sont analysés AWS et pris en charge par un AWS partenaire indépendant.
  - Utilisez un module complémentaire Marketplace pour ajouter des fonctionnalités précieuses et sophistiquées à votre cluster, telles que la surveillance avec Splunk.
  - Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires Marketplace".
- Modules complémentaires communautaires : Ces modules complémentaires sont analysés par la communauté open source AWS mais pris en charge par celle-ci.
  - Utilisez un module complémentaire communautaire pour réduire la complexité de l'installation de logiciels open source courants, tels que Kubernetes Metrics Server.
  - Les modules complémentaires communautaires ne sont pas créés par AWS. AWS valide uniquement les modules complémentaires communautaires pour la compatibilité des versions.
  - Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires communautaires".

Le tableau suivant détaille l'étendue du support pour chaque type de module complémentaire :

Catégorie	Fonctionnalité	AWS modules complémentaires	AWS Modules complémentaires Marketplace	Extensions communautaires
Développement	Construit par AWS	Oui	Non	Oui
Développement	Validé par AWS	Oui	Non	Oui*
Développement	Validé par le AWS partenaire	Non	Oui	Non
Maintenance	Scanné par AWS	Oui	Oui	Oui
Maintenance	Corrigé par AWS	Oui	Non	Oui
Maintenance	Corrigé par un partenaire AWS	Non	Oui	Non
Distribution	Publié par AWS	Oui	Non	Oui

Catégorie	Fonctionnalité	AWS modules complémentaires	AWS Modules complémentaires Marketplace	Extensions communautaires
Distribution	Publié par AWS Partner	Non	Oui	Non
Support	Support d'installation de base par AWS	Oui	Oui	Oui
Support	AWS Support complet	Oui	Non	Non
Support	Support complet AWS pour les partenaires	Non	Oui	Non

*: La validation des modules complémentaires communautaires inclut uniquement la compatibilité des versions de Kubernetes. Par exemple, si vous installez un module complémentaire communautaire sur un cluster, vérifiez AWS s'il est compatible avec la version Kubernetes de votre cluster.

AWS Les modules complémentaires Marketplace peuvent télécharger des dépendances logicielles supplémentaires à partir de sources externes extérieures à AWS. Ces dépendances externes ne sont ni scannées ni validées par AWS. Tenez compte de vos exigences en matière de sécurité lorsque vous déployez des modules complémentaires AWS Marketplace qui récupèrent des dépendances externes.

AWS modules complémentaires

Les modules complémentaires Amazon EKS suivants peuvent être créés sur votre cluster. Vous pouvez consulter la liste la plus récente des modules complémentaires disponibles à l'aide `eksctl` de la AWS Management Console CLI ou de la AWS CLI. Pour voir tous les modules complémentaires disponibles ou pour installer un module complémentaire, consultez the section called "Création d'un module complémentaire". Si un module complémentaire nécessite des autorisations IAM, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section

called "Fournisseur IAM OIDC". Vous pouvez créer ou supprimer un module complémentaire après l'avoir installé. Pour plus d'informations, consultez the section called "Mettre à jour un module complémentaire" ou the section called "Supprimer un module complémentaire". Pour plus d'informations sur les considérations spécifiques à l'exécution de modules complémentaires EKS avec Amazon EKS Hybrid Nodes, consultezthe section called "Configuration des modules complémentaires".

Vous pouvez utiliser l'un des modules complémentaires Amazon EKS suivants.

Description	En savoir plus	Types de calcul compatibles
Fournissez un réseau VPC natif à votre cluster	the section called "Plugin CNI Amazon VPC pour Kubernetes"	EC2
Un serveur DNS flexible et extensible qui peut servir de DNS du cluster Kubernetes	the section called "CoreDNS"	EC2, Fargate, mode automatique EKS, nœuds hybrides EKS
Maintenir les règles du réseau sur chaque EC2 nœud Amazon	the section called "`Kube-proxy` "	EC2, Nœuds hybrides EKS
Fournissez un espace de stockage Amazon EBS pour votre cluster	the section called "Pilote CSI Amazon EBS"	EC2
Fournissez un espace de stockage Amazon EFS pour votre cluster	the section called "Pilote CSI Amazon EFS"	EC2, Mode automatique EKS
Fournissez un espace de stockage Amazon FSx for Lustre pour votre cluster	the section called "pilote Amazon FSx CSI"	EC2, Mode automatique EKS
Fournissez un espace de stockage Amazon S3 pour votre cluster	the section called "Pilote CSI Mountpoint pour Amazon S3"	EC2, Mode automatique EKS

Description	En savoir plus	Types de calcul compatibles
Détectez d'autres problèmes de santé des nœuds	the section called "Agent de surveillance des nœuds"	EC2, Nœuds hybrides EKS
Activez l'utilisation de la fonctionnalité de capture instantanée dans les pilotes CSI compatibles, tels que le pilote Amazon EBS CSI	the section called "Contrôleur d'instantané CSI"	EC2, Fargate, mode automatique EKS, nœuds hybrides EKS
SageMaker HyperPod la gouvernance des tâches optimise l'allocation et l'utilisation des ressources informatiques entre les équipes dans les clusters Amazon EKS, remédiant ainsi aux inefficiences en matière de priorisation des tâches et de partage des ressources.	the section called "Gouvernance des SageMaker HyperPod tâches Amazon"	EC2, Mode automatique EKS,
Amazon SageMaker HyperPod Observability AddOn fournit des fonctionnalités complètes de surveillance et d'observabilité pour HyperPod les clusters.	the section called "Module complémentaire Amazon SageMaker HyperPod Observability"	EC2, Mode automatique EKS,
Un agent Kubernetes qui collecte et transmet les données de flux réseau à Amazon CloudWatch, permettant ainsi une surveillance complète des connexions TCP entre les nœuds du cluster.	the section called "AWS Agent de surveillance du flux réseau"	EC2, Mode automatique EKS

Description	En savoir plus	Types de calcul compatibles
Distribution sécurisée, prête pour la production et AWS prise en charge du projet OpenTelemetry	the section called "AWS Distro pour OpenTelemetry"	EC2, Fargate, mode automatiq ue EKS, nœuds hybrides EKS
Service de surveillance de la sécurité qui analyse et traite les sources de données de base, notamment les événements AWS CloudTrail de gestion et les journaux de flux Amazon VPC. Amazon traite GuardDuty également des fonctionnalités, telles que les journaux d'audit Kubernete s et la surveillance du temps d'exécution	the section called " GuardDuty Agent Amazon"	EC2, Mode automatique EKS
Service de surveillance et d'observabilité fourni par AWS. Ce module complémentaire installe l' CloudWatch agent et active à la fois CloudWatc h Application Signals et CloudWatch Container Insights avec une observabilité améliorée pour Amazon EKS	the section called "Agent Amazon CloudWatch Observability"	EC2, mode automatique EKS, nœuds hybrides EKS
Possibilité de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance fournissent des informations d'identification aux EC2 instances	the section called "Agent d'identité du pod EKS"	EC2, Nœuds hybrides EKS

Description	En savoir plus	Types de calcul compatibles
Activez cert-manager pour émettre des certificats X.509 à partir de AWS Private CA. Nécessite un gestionnaire de certificats.	the section called "AWS Connecteur CA privé pour Kubernetes"	EC2, Fargate, mode automatiq ue EKS, nœuds hybrides EKS
Générez des métriques Prometheus sur les performan ces des appareils réseau SR-IOV	the section called "Exportateur de métriques réseau SR-IOV"	EC2

Plugin CNI Amazon VPC pour Kubernetes

Le plug-in Amazon VPC CNI pour Kubernetes Le module complémentaire Amazon EKS est un plug-in d'interface réseau de conteneurs (CNI) Kubernetes qui fournit un réseau VPC natif à votre cluster. Le type autogéré ou géré de ce module complémentaire est installé sur chaque EC2 nœud Amazon, par défaut. Pour plus d'informations, consultez la section Plug-in CNI (Kubernetes Container Network Interface).

> **Note**
>
> Il n'est pas nécessaire d'installer ce module complémentaire sur les clusters Amazon EKS Auto Mode. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique d'Amazon EKS".

Le nom du module complémentaire Amazon EKS est`vpc-cni`.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA".

Si votre cluster utilise la famille `IPv4`, les autorisations définies dans la politique AmazonEKS_CNI_Policy sont requises. Si votre cluster utilise cette `IPv6` famille, vous devez créer

une politique IAM avec les autorisations en IPv6 mode. Vous pouvez créer un rôle IAM, y associer l'une des politiques et annoter le compte de service Kubernetes utilisé par le module complémentaire à l'aide de la commande suivante.

Remplacez *my-cluster* par le nom de votre cluster et *AmazonEKSVPCCNIRole* par le nom que vous souhaitez pour votre rôle. Si votre cluster utilise la famille IPv6, remplacez *AmazonEKS_CNI_Policy* par le nom de la politique que vous avez créée. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle, y associer la politique et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

Informations de mise à jour

Vous ne pouvez mettre à jour qu'une seule version mineure à la fois. Par exemple, si votre version actuelle est `1.28.x-eksbuild.y` et que vous voulez la mettre à jour vers `1.30.x-eksbuild.y`, vous devez d'abord mettre à jour votre version actuelle vers `1.29.x-eksbuild.y` et ensuite la mettre à nouveau à jour vers `1.30.x-eksbuild.y`. Pour plus d'informations sur la mise à jour du module complémentaire, consultez la rubrique the section called "Mise à jour (module complémentaire EKS)".

CoreDNS

Le module complémentaire CoreDNS Amazon EKS est un serveur DNS flexible et extensible qui peut servir de DNS du cluster Kubernetes. Le type autogéré ou géré de ce module complémentaire a été installé par défaut lorsque vous avez créé votre cluster. Lorsque vous lancez un cluster Amazon EKS avec au moins un nœud, deux réplicas de l'image CoreDNS sont déployés par défaut, quel que soit le nombre de nœuds déployés dans votre cluster. Les pods CoreDNS fournissent une résolution de noms pour tous les pods du cluster. Vous pouvez déployer les pods CoreDNS sur les nœuds Fargate si votre cluster inclut un profil Fargate avec un espace de noms qui correspond à l'espace de noms pour le déploiement de CoreDNS. Pour de plus amples informations, consultez the section called "Définir des profils".

> **Note**
> Il n'est pas nécessaire d'installer ce module complémentaire sur les clusters Amazon EKS Auto Mode. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique d'Amazon EKS".

Le nom du module complémentaire Amazon EKS est`coredns`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour en savoir plus sur CoreDNS, consultez les sections Utilisation de CoreDNS pour la découverte de services et personnalisation du service DNS dans la documentation de Kubernetes.

`Kube-proxy`

Le module complémentaire `Kube-proxy` Amazon EKS gère les règles réseau sur chaque EC2 nœud Amazon. Il permet la communication réseau avec vos Pods. Le type autogéré ou géré de ce module complémentaire est installé par défaut sur chaque EC2 nœud Amazon de votre cluster.

> **Note**
> Il n'est pas nécessaire d'installer ce module complémentaire sur les clusters Amazon EKS Auto Mode. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique d'Amazon EKS".

Le nom du module complémentaire Amazon EKS est`kube-proxy`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations de mise à jour

Avant de mettre à jour votre version actuelle, tenez compte des exigences suivantes :

- `Kube-proxy`sur un cluster Amazon EKS a la même politique de compatibilité et de distorsion que Kubernetes.

Informations supplémentaires

Pour en savoir plus`kube-proxy`, consultez kube-proxy dans la documentation de Kubernetes.

Pilote CSI Amazon EBS

Le pilote Amazon EBS CSI (module complémentaire Amazon EKS) est un plug-in Kubernetes Container Storage Interface (CSI) qui fournit un stockage Amazon EBS pour votre cluster.

> **Note**
>
> Il n'est pas nécessaire d'installer ce module complémentaire sur les clusters Amazon EKS Auto Mode. Le mode automatique inclut une fonctionnalité de stockage par blocs. Pour de plus amples informations, veuillez consulter the section called "Déployez une charge de travail dynamique".

Le nom du module complémentaire Amazon EKS est`aws-ebs-csi-driver`.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA". Les autorisations définies dans la EBSCSIDriverpolitique AWS gérée par Amazon Policy sont obligatoires. Vous pouvez créer un rôle IAM et y attacher la politique gérée à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_EBS_CSI_DriverRole`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil ou une clé KMS personnalisée pour le chiffrement, reportez-vous à la section the section called "Étape 1 : Créer un rôle IAM".

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
```

```
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
    --approve
```

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultezthe section called "Amazon EBS".

Pilote CSI Amazon EFS

Le pilote Amazon EFS CSI (module complémentaire Amazon EKS) est un plugin Kubernetes Container Storage Interface (CSI) qui fournit un stockage Amazon EFS pour votre cluster.

Le nom du module complémentaire Amazon EKS est`aws-efs-csi-driver`.

Autorisations IAM requises

Autorisations IAM requises — Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA". Les autorisations définies dans la EFSCSIDriverpolitique AWS gérée par Amazon Policy sont obligatoires. Vous pouvez créer un rôle IAM et y attacher la politique gérée à l'aide des commandes suivantes. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_EFS_CSI_DriverRole`* par le nom que vous souhaitez pour votre rôle. Ces commandes nécessitent que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil, voir the section called "Étape 1 : Créer un rôle IAM".

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
```

```
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultezthe section called “Amazon EFS”.

pilote Amazon FSx CSI

Le module complémentaire Amazon EKS du pilote Amazon FSx CSI est un plugin Kubernetes Container Storage Interface (CSI) qui fournit le stockage Amazon FSx for Lustre pour votre cluster.

Le nom du module complémentaire Amazon EKS est`aws-fsx-csi-driver`.

> **Note**
>
> - Les installations préexistantes du pilote Amazon FSx CSI dans le cluster peuvent provoquer l'échec de l'installation des modules complémentaires. Lorsque vous tentez d'installer la version complémentaire Amazon EKS alors qu'un pilote FSx CSI autre qu'EKS existe, l'installation échoue en raison de conflits de ressources. Utilisez le `OVERWRITE` drapeau lors de l'installation pour résoudre ce problème :
>
>  ```
>  aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>  --resolve-conflicts OVERWRITE
>  ```
>
> - Le module complémentaire Amazon FSx CSI Driver EKS nécessite l'agent EKS Pod Identity pour l'authentification. Sans ce composant, le module complémentaire échouera avec l'erreur`Amazon EKS Pod Identity agent is not installed in the cluster`, empêchant ainsi les opérations de volume. Installez l'agent Pod Identity avant ou après le déploiement du module complémentaire FSx CSI Driver. Pour de plus amples informations, veuillez consulter the section called “Configuration de l'agent”.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called “Identifiants auprès de l'IRSA”. Les autorisations définies dans la politique FSx FullAccess AWS gérée par Amazon sont obligatoires. Vous pouvez créer un rôle IAM et y attacher la politique gérée

à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster et *AmazonEKS_FSx_CSI_DriverRole* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil.

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultezthe section called "Amazon FSx pour Lustre".

Pilote CSI Mountpoint pour Amazon S3

Le module complémentaire Amazon EKS du pilote CSI Mountpoint pour Amazon S3 est un plugin Kubernetes Container Storage Interface (CSI) qui fournit un stockage Amazon S3 à votre cluster.

Le nom du module complémentaire Amazon EKS est`aws-mountpoint-s3-csi-driver`.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA".

Le rôle IAM créé nécessitera une stratégie d'accès à S3. Suivez les recommandations relatives aux autorisations IAM de Mountpoint lors de la création de la politique. Vous pouvez également utiliser le lien de politique AWS gérée :iam/home?#/policies/arn:aws: iam : :aws:policy/AmazonS3 FullAccess $JSONEditor [AmazonS3, type="console "]FullAccess, mais cette politique gérée accorde plus d'autorisations que ce qui est nécessaire pour Mountpoint.

Vous pouvez créer un rôle IAM et y attacher votre politique à l'aide des commandes suivantes. Remplacez-le *my-cluster* par le nom de votre cluster, *region-code* par le code de AWS région correct, *AmazonEKS_S3_CSI_DriverRole* par le nom de votre rôle et *AmazonEKS_S3_CSI_DriverRole_ARN* par l'ARN du rôle. Ces commandes nécessitent que eksctl

soit installé sur votre appareil. Pour obtenir des instructions sur l'utilisation de la console ou de la AWS CLI IAM, consultezthe section called "Étape 2 : Créer un rôle IAM".

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultezthe section called "Mountpoint pour Amazon S3".

Contrôleur d'instantané CSI

Le contrôleur de capture instantanée CSI (Container Storage Interface) permet d'utiliser la fonctionnalité de capture instantanée dans les pilotes CSI compatibles, tels que le pilote Amazon EBS CSI.

Le nom du module complémentaire Amazon EKS est`snapshot-controller`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultezthe section called "Contrôleur d'instantané CSI".

Gouvernance des SageMaker HyperPod tâches Amazon

SageMaker HyperPod la gouvernance des tâches est un système de gestion robuste conçu pour rationaliser l'allocation des ressources et garantir une utilisation efficace des ressources

informatiques au sein des équipes et des projets pour vos clusters Amazon EKS. Cela permet aux administrateurs de définir :

- Niveaux de priorité pour différentes tâches
- Calculez l'allocation pour chaque équipe
- Comment chaque équipe prête et emprunte le calcul inutilisé
- Si une équipe préempte ses propres tâches

HyperPod la gouvernance des tâches fournit également l'observabilité du cluster Amazon EKS, offrant une visibilité en temps réel sur la capacité du cluster. Cela inclut la disponibilité et l'utilisation du calcul, la répartition et l'utilisation des équipes, ainsi que les informations sur l'exécution des tâches et les temps d'attente, vous permettant ainsi de prendre des décisions éclairées et de gérer les ressources de manière proactive.

Le nom du module complémentaire Amazon EKS est`amazon-sagemaker-hyperpod-taskgovernance`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultez la section Gouvernance des SageMaker HyperPod tâches

Module complémentaire Amazon SageMaker HyperPod Observability

Le module complémentaire Amazon SageMaker HyperPod Observability fournit des fonctionnalités complètes de surveillance et d'observabilité pour HyperPod les clusters. Ce module complémentaire déploie et gère automatiquement les composants de surveillance essentiels, notamment l'exportateur de nœuds, l'exportateur DCGM et l'exportateur EFA. kube-state-metrics Il collecte et transmet les métriques à une instance Amazon Managed Prometheus (AMP) désignée par le client et expose un point de terminaison OTLP pour les métriques personnalisées et l'ingestion d'événements liés aux tâches de formation des clients.

Le module complémentaire s'intègre à l'ensemble de l' HyperPod écosystème en extrayant les métriques de divers composants, notamment le module complémentaire HyperPod Task

Governance, HyperPod Training Operator, Kubeflow et KEDA. Toutes les mesures collectées sont centralisées dans Amazon Managed Prometheus, ce qui permet aux clients d'obtenir une vue unifiée de l'observabilité via les tableaux de bord Amazon Managed Grafana. Cela fournit une end-to-end visibilité sur l'état du cluster, l'utilisation des ressources et les performances des tâches de formation dans l'ensemble de HyperPod l'environnement.

Le nom du module complémentaire Amazon EKS est`amazon-sagemaker-hyperpod-observability`.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA". Les politiques gérées suivantes sont requises :

- `AmazonPrometheusRemoteWriteAccess`- pour écrire à distance des métriques du cluster vers AMP
- `CloudWatchAgentServerPolicy`- pour écrire à distance les journaux du cluster vers CloudWatch

Informations supplémentaires

Pour en savoir plus sur le module complémentaire et ses fonctionnalités, consultez SageMaker HyperPod Observabilité.

AWS Agent de surveillance du flux réseau

L'agent Amazon CloudWatch Network Flow Monitor est une application Kubernetes qui collecte les statistiques de connexion TCP à partir de tous les nœuds d'un cluster et publie des rapports sur le flux réseau dans Amazon CloudWatch Network Flow Monitor Ingestion. APIs

Le nom du module complémentaire Amazon EKS est`aws-network-flow-monitoring-agent`.

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM.

Vous devez associer la politique `CloudWatchNetworkFlowMonitorAgentPublishPolicy` gérée au module complémentaire.

Pour plus d'informations sur la configuration IAM requise, consultez la politique IAM sur le référentiel d'agents GitHub Amazon CloudWatch Network Flow Monitor.

Pour plus d'informations sur la politique gérée, consultez CloudWatchNetworkFlowMonitorAgentPublishPolicyle guide de CloudWatch l'utilisateur Amazon.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultez le GitHub dépôt de l'agent Amazon CloudWatch Network Flow Monitor.

Agent de surveillance des nœuds

Le module complémentaire Amazon EKS de l'agent de surveillance des nœuds peut détecter d'autres problèmes de santé des nœuds. Ces signaux de santé supplémentaires peuvent également être exploités par la fonction optionnelle de réparation automatique des nœuds pour remplacer automatiquement les nœuds selon les besoins.

> **Note**
>
> Il n'est pas nécessaire d'installer ce module complémentaire sur les clusters Amazon EKS Auto Mode. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique d'Amazon EKS".

Le nom du module complémentaire Amazon EKS est`eks-node-monitoring-agent`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation supplémentaire.

Informations supplémentaires

Pour de plus amples informations, veuillez consulter the section called "Santé des nœuds".

AWS Distro pour OpenTelemetry

Le module complémentaire AWS Distro for OpenTelemetry Amazon EKS est une distribution sécurisée, prête pour la production et AWS prise en charge du projet. OpenTelemetry Pour plus d'informations, consultez AWS Distro for OpenTelemetry on GitHub.

Le nom du module complémentaire Amazon EKS est`adot`.

Autorisations IAM requises

Ce module complémentaire ne nécessite des autorisations IAM que si vous utilisez l'une des ressources personnalisées préconfigurées auxquelles vous pouvez accéder via une configuration avancée.

Informations supplémentaires

Pour plus d'informations, consultez Getting Started with AWS Distro pour OpenTelemetry l'utilisation des modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

ADOT exige que le `cert-manager` module complémentaire soit déployé sur le cluster comme condition préalable, sinon ce module complémentaire ne fonctionnera pas s'il est déployé directement à l'aide du https://registry.terraform. io/modules/terraform-aws-modules/eks/aws/latest`cluster_addons`propriété. Pour plus d'informations sur les exigences, reportez-vous à la section Conditions requises pour démarrer avec AWS Distro pour OpenTelemetry l'utilisation des modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

GuardDuty Agent Amazon

Le module complémentaire Amazon EKS de l' GuardDuty agent Amazon collecte les événements d'exécution (accès aux fichiers, exécution des processus, connexions réseau) à partir des nœuds de votre cluster EKS pour les analyser par GuardDuty Runtime Monitoring. GuardDuty lui-même (et non l'agent) est le service de surveillance de la sécurité qui analyse et traite les sources de données fondamentales, notamment les événements de AWS CloudTrail gestion et les journaux de flux Amazon VPC, ainsi que des fonctionnalités telles que les journaux d'audit Kubernetes et la surveillance du temps d'exécution.

Le nom du module complémentaire Amazon EKS est`aws-guardduty-agent`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour plus d'informations, consultez la section Surveillance du temps d'exécution pour les clusters Amazon EKS sur Amazon GuardDuty.

- Pour détecter les menaces de sécurité potentielles dans vos clusters Amazon EKS, activez la surveillance de l' GuardDuty exécution Amazon et déployez l'agent de GuardDuty sécurité sur vos clusters Amazon EKS.

Agent Amazon CloudWatch Observability

L'agent Amazon CloudWatch Observability (Amazon EKS) ajoute le service de surveillance et d'observabilité fourni par. AWS Ce module complémentaire installe l' CloudWatch agent et active à la fois CloudWatch Application Signals et CloudWatch Container Insights avec une observabilité améliorée pour Amazon EKS. Pour plus d'informations, consultez Amazon CloudWatch Agent.

Le nom du module complémentaire Amazon EKS est`amazon-cloudwatch-observability`.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Identifiants auprès de l'IRSA". Les autorisations indiquées dans les politiques gérées link :iam/home#/policies/ arn:aws: iam : :aws :policy/AWSXrayWriteOnlyAccess[AWSXrayWriteOnlyAccess,type="console"] and link:iam/home#/policies/arn:aws: iam : :aws:policy/ CloudWatchAgentServerPolicy [CloudWatchAgentServerPolicy, type="console "] sont obligatoires. AWS Vous pouvez créer un rôle IAM, y associer les politiques gérées et annoter le compte de service Kubernetes utilisé par le module complémentaire à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_Observability_role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle, y associer la politique et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws: iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

Informations supplémentaires

Pour plus d'informations, consultez la section Installation de l' CloudWatch agent.

AWS Connecteur CA privé pour Kubernetes

Le connecteur AWS Private CA pour Kubernetes est un module complémentaire pour cert-manager qui permet aux utilisateurs d'obtenir des certificats auprès d'une autorité de certification privée (autorité de certification AWS privée).AWS

- Le nom du module complémentaire Amazon EKS est`aws-privateca-connector-for-kubernetes`.
- L'espace de noms du module complémentaire est`aws-privateca-issuer`.

Ce module complémentaire nécessite`cert-manager`. `cert-manager`est disponible sur Amazon EKS en tant que module complémentaire communautaire. Pour plus d'informations sur ce module complémentaire, consultezthe section called "Responsable des certificats". Pour plus d'informations sur l'installation de modules complémentaires, consultezthe section called "Création d'un module complémentaire".

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM.

Utilisez EKS Pod Identities pour associer la politique `AWSPrivateCAConnectorForKubernetesPolicy` IAM au compte de service `aws-privateca-issuer` Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Utiliser Pod Identities".

Pour plus d'informations sur les autorisations requises, consultez le AWSPrivateCAConnectorForKubernetesPolicymanuel AWS Managed Policy Reference.

Informations supplémentaires

Pour plus d'informations, consultez le référentiel AWS Private CA Issuer for Kubernetes GitHub .

Pour plus d'informations sur la configuration du module complémentaire, consultez values.yaml dans le dépôt. `aws-privateca-issuer` GitHub Vérifiez que la version de values.yaml correspond à la version du module complémentaire installé sur votre cluster.

Ce module complémentaire tolère les `CriticalAddonsOnly` taches produites par le mode automatique `system` NodePool d'EKS. Pour de plus amples informations, veuillez consulter the section called "Exécutez des modules complémentaires essentiels".

Agent d'identité du pod EKS

Le module complémentaire Amazon EKS Pod Identity Agent Amazon EKS permet de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance fournissent des informations d'identification aux EC2 instances.

> **Note**
>
> Il n'est pas nécessaire d'installer ce module complémentaire sur les clusters Amazon EKS Auto Mode. Le mode automatique Amazon EKS s'intègre à EKS Pod Identity. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique d'Amazon EKS".

Le nom du module complémentaire Amazon EKS est`eks-pod-identity-agent`.

Autorisations IAM requises

Le module complémentaire Pod Identity Agent lui-même ne nécessite pas de rôle IAM. Il utilise les autorisations du rôle IAM du nœud Amazon EKS pour fonctionner, mais il n'a pas besoin d'un rôle IAM dédié pour le module complémentaire.

Informations de mise à jour

Vous ne pouvez mettre à jour qu'une seule version mineure à la fois. Par exemple, si votre version actuelle est `1.28.x-eksbuild.y` et que vous voulez la mettre à jour vers `1.30.x-eksbuild.y`, vous devez d'abord mettre à jour votre version actuelle vers `1.29.x-eksbuild.y` et ensuite la mettre à nouveau à jour vers `1.30.x-eksbuild.y`. Pour plus d'informations sur la mise à jour du module complémentaire, consultez la rubrique the section called "Mettre à jour un module complémentaire".

Exportateur de métriques réseau SR-IOV

Le module complémentaire Amazon EKS SR-IOV Network Metrics Exporter collecte et expose les métriques relatives aux appareils réseau SR-IOV au format Prometheus. Il permet de surveiller les

performances du réseau SR-IOV sur les nœuds bare metal d'EKS. L'exportateur fonctionne comme un nœud DaemonSet doté d'interfaces SR-IOV-capable réseau et exporte des métriques qui peuvent être récupérées par Prometheus.

> **Note**
>
> Ce module complémentaire nécessite des nœuds dotés d'interfaces SR-IOV-capable réseau.

Propriété	Valeur
Nom du module	`sriov-network-metrics-exporter`
Espace de noms	`monitoring`
Documentation	Dépôt de l'exportateur de métriques réseau SR-IOV GitHub
Nom du compte de service	Aucun
Politique IAM gérée	Aucun
Autorisations IAM personnalisées	Aucun

Modules complémentaires communautaires

Vous pouvez l'utiliser AWS APIs pour installer des modules complémentaires communautaires, tels que le serveur Kubernetes Metrics. Vous pouvez choisir d'installer des modules complémentaires communautaires en tant que modules complémentaires Amazon EKS afin de réduire la complexité de la maintenance du logiciel sur plusieurs clusters.

Par exemple, vous pouvez utiliser l' AWS API, la CLI ou la console de gestion pour installer des modules complémentaires communautaires. Vous pouvez installer un module complémentaire communautaire lors de la création du cluster.

Vous gérez les modules complémentaires communautaires de la même manière que les modules complémentaires Amazon EKS existants. Les modules complémentaires communautaires sont différents des modules existants en ce sens qu'ils ont une portée de support unique.

> **Note**
>
> L'utilisation des modules complémentaires communautaires est à votre discrétion. Dans le cadre du modèle de responsabilité partagée entre vous et AWS, vous êtes censé comprendre ce que vous installez dans votre cluster pour ces plugins tiers. Vous êtes également responsable des modules complémentaires communautaires qui répondent aux besoins de sécurité de votre cluster. Pour de plus amples informations, veuillez consulter the section called "Support pour les logiciels déployés sur EKS".

Les modules complémentaires communautaires ne sont pas créés par AWS. AWS valide uniquement les modules complémentaires communautaires pour la compatibilité des versions. Par exemple, si vous installez un module complémentaire communautaire sur un cluster, vérifiez AWS s'il est compatible avec la version Kubernetes de votre cluster.

Il est important de noter AWS qu'il ne fournit pas un support complet pour les modules complémentaires communautaires. AWS prend uniquement en charge les opérations de cycle de vie effectuées à l'aide de modules complémentaires AWS APIs, telles que l'installation ou la suppression de modules complémentaires.

Si vous avez besoin d'assistance pour un module complémentaire communautaire, utilisez les ressources du projet existantes. Par exemple, vous pouvez créer un GitHub problème sur le dépôt du projet.

Déterminer le type de module complémentaire

Vous pouvez utiliser la AWS CLI pour déterminer le type d'un module complémentaire Amazon EKS.

Utilisez la commande CLI suivante pour récupérer les informations relatives à un module complémentaire. Vous pouvez le remplacer `metrics-server` par le nom de n'importe quel module complémentaire.

```
aws eks describe-addon-versions --addon-name metrics-server
```

Vérifiez la sortie de la CLI pour le `owner` champ.

```
{
    "addons": [
        {
```

```
            "addonName": "metrics-server",
            "type": "observability",
            "owner": "community",
            "addonVersions": [
```

Si la valeur de `owner` est`community`, le module complémentaire est un module complémentaire communautaire. AWS fournit uniquement un support pour l'installation, la mise à jour et la suppression du module complémentaire. Si vous avez des questions concernant les fonctionnalités et le fonctionnement du module complémentaire lui-même, utilisez les ressources communautaires telles que GitHub les problèmes.

Installer ou mettre à jour le module complémentaire communautaire

Vous installez ou mettez à jour les modules complémentaires communautaires de la même manière que les autres modules complémentaires Amazon EKS.

- the section called "Création d'un module complémentaire"
- the section called "Mettre à jour un module complémentaire"
- the section called "Supprimer un module complémentaire"

Extensions communautaires disponibles

Les modules complémentaires communautaires suivants sont disponibles sur Amazon EKS.

- the section called "Serveur de métriques Kubernetes"
- the section called "kube-state-metrics"
- the section called "Prometheus Node Exporter"
- the section called "Responsable des certificats"
- the section called "DNS externe"
- the section called "Fluent Bit"

Serveur de métriques Kubernetes

Le serveur de métriques Kubernetes est une source évolutive et efficace de métriques de ressources de conteneur pour les pipelines de mise à l'échelle automatique intégrés de Kubernetes. Il collecte les métriques des ressources à partir de Kubelets et les expose dans le serveur d'API Kubernetes via l'API Metrics pour une utilisation par Horizontal Pod Autoscaler et Vertical Pod Autoscaler.

Propriété	Valeur
Nom du module	`metrics-server`
Espace de noms	`kube-system`
Documentation	GitHub Readme
Nom du compte de service	Aucun
Politique IAM gérée	Aucun
Autorisations IAM personnalisées	Aucun

kube-state-metrics

Agent complémentaire pour générer et exposer des métriques au niveau du cluster.

L'état des objets Kubernetes dans l'API Kubernetes peut être exposé sous forme de métriques. Un agent complémentaire appelé kube-state-metrics peut se connecter au serveur d'API Kubernetes et exposer un point de terminaison HTTP avec des métriques générées à partir de l'état des objets individuels du cluster. Il expose diverses informations sur l'état des objets, telles que les étiquettes et les annotations, les heures de démarrage et de fin, le statut ou la phase dans laquelle se trouve actuellement l'objet.

Propriété	Valeur
Nom du module	`kube-state-metrics`
Espace de noms	`kube-state-metrics`
Documentation	Métriques relatives aux états des objets Kubernetes dans Kubernetes Docs
Nom du compte de service	Aucun
Politique IAM gérée	Aucun
Autorisations IAM personnalisées	Aucun

Prometheus Node Exporter

Exportateur Prometheus pour les métriques matérielles et de système d'exploitation exposées par les noyaux *NIX, écrit en Go avec des collecteurs de métriques enfichables. Le Prometheus Node Exporter expose une grande variété de métriques liées au matériel et au noyau.

Propriété	Valeur
Nom du module	`prometheus-node-exporter`
Espace de noms	`prometheus-node-exporter`
Documentation	Surveillance des métriques des hôtes Linux avec le Node Exporter dans Prometheus Docs
Nom du compte de service	Aucun
Politique IAM gérée	Aucun
Autorisations IAM personnalisées	Aucun

Responsable des certificats

Cert Manager peut être utilisé pour gérer la création et le renouvellement des certificats.

Propriété	Valeur
Nom du module	`cert-manager`
Espace de noms	`cert-manager`
Documentation	Documents relatifs au gestionnaire de certificats
Nom du compte de service	Aucun
Politique IAM gérée	Aucun
Autorisations IAM personnalisées	Aucun

DNS externe

Le module complémentaire EKS DNS externe peut être utilisé pour gérer les enregistrements DNS Route53 via les ressources Kubernetes.

Les autorisations DNS externes peuvent être réduites aux `route53:ChangeResourceRecordSets` zones hébergées que vous souhaitez gérer et `route53:ListResourceRecordSets` sur celles-ci. `route53:ListHostedZones`

Propriété	Valeur
Nom du module	`external-dns`
Espace de noms	`external-dns`
Documentation	GitHub Readme
Nom du compte de service	`external-dns`
Politique IAM gérée	`arn:aws: iam::aws:policy/AmazonRoute53FullAccess`
Autorisations IAM personnalisées	Aucun

Fluent Bit

Fluent Bit est un processeur de log et un redirecteur légers et performants. Il vous permet data/logs de collecter des données auprès de différentes sources, de les unifier et de les envoyer vers plusieurs destinations, notamment Amazon CloudWatch Logs, Amazon S3 et Amazon Kinesis Data Firehose. Fluent Bit est conçu dans un souci de performance et d'efficacité des ressources, ce qui le rend idéal pour les environnements Kubernetes.

Ce module complémentaire ne nécessite pas d'autorisations IAM dans la configuration par défaut. Toutefois, vous devrez peut-être accorder à ce module complémentaire des autorisations IAM si vous configurez un emplacement AWS de sortie. Pour de plus amples informations, veuillez consulter the section called "Utiliser Pod Identities".

Propriété	Valeur
Nom du module	`fluent-bit`
Espace de noms	`fluent-bit`
Documentation	Documentation Fluent Bit
Nom du compte de service	`fluent-bit`
Politique IAM gérée	Aucun
Autorisations IAM personnalisées	Aucun

Afficher les attributions

Vous pouvez télécharger les attributions open source et les informations de licence pour les modules complémentaires communautaires.

1. Déterminez le nom et la version du module complémentaire pour lequel vous souhaitez télécharger les attributions.
2. Mettez à jour la commande suivante avec le nom et la version :

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
   ```

   Par exemple :

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/v2.14.0-eksbuild.1/attributions.zip
   ```

3. Utilisez la commande pour télécharger le fichier.

Utilisez ce fichier zip pour consulter les informations relatives aux attributions de licence.

AWS Modules complémentaires Marketplace

Outre la liste précédente de modules complémentaires Amazon EKS, vous pouvez également ajouter une large sélection de modules complémentaires Amazon EKS pour logiciels opérationnels proposés

par des fournisseurs indépendants de logiciels. Sélectionnez un module complémentaire pour en savoir plus à son sujet et sur les conditions d'installation de celui-ci.

Accuknox

Le nom du module complémentaire est `accuknox_kubearmor` et l'espace de noms est`kubearmor`. Accuknox publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Getting Started with KubeArmor dans la KubeArmor documentation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Akuity

Le nom du module complémentaire est `akuity_agent` et l'espace de noms est`akuity`. Akuity publie le module complémentaire.

Pour plus d'informations sur le fonctionnement du module complémentaire, consultez la section Installation de l'agent Akuity sur Amazon EKS avec le module complémentaire Akuity EKS dans la documentation de la plateforme Akuity.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Calyptia

Le nom du module complémentaire est `calyptia_fluent-bit` et l'espace de noms est`calytia-fluentbit`. Calyptia publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Getting Started with Calyptia Core Agent sur le site Web de documentation de Calyptia.

Nom du compte de service

Le nom du compte de service est`clyptia-fluentbit`.

AWS politique IAM gérée

Ce module complémentaire utilise la politique `AWSMarketplaceMeteringRegisterUsage` gérée. Pour plus d'informations, consultez AWSMarketplaceMeteringRegisterUsagele Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *`my-cluster`* par le nom de votre cluster et *`my-calyptia-role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

Collecteur d'observabilité Cisco

Le nom du module complémentaire est `cisco_cisco-cloud-observability-collectors` et l'espace de noms est`appdynamics`. Cisco publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, voir Utiliser les modules complémentaires Cisco Cloud Observability AWS Marketplace dans la AppDynamics documentation Cisco.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Opérateur d'observabilité Cisco

Le nom du module complémentaire est `cisco_cisco-cloud-observability-operators` et l'espace de noms est`appdynamics`. Cisco publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, voir Utiliser les modules complémentaires Cisco Cloud Observability AWS Marketplace dans la AppDynamics documentation Cisco.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

CLOUDSOFT

Le nom du module complémentaire est `cloudsoft_cloudsoft-amp` et l'espace de noms est`cloudsoft-amp`. CLOUDSOFT publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Amazon EKS ADDON dans la documentation CLOUDSOFT.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Berceau

Le nom du module complémentaire est `cribl_cribledge` et l'espace de noms est`cribledge`. Cribl publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du module complémentaire Cribl Amazon EKS pour Edge dans la documentation Cribl

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Dynatrace

Le nom du module complémentaire est `dynatrace_dynatrace-operator` et l'espace de noms est`dynatrace`. Dynatrace publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Surveillance de Kubernetes dans la documentation de dynatrace.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Arbre dattier

Le nom du module complémentaire est `datree_engine-pro` et l'espace de noms est`datree`. Datree publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Amazon EKS Integration dans la documentation de Datree.

Nom du compte de service

Le nom du compte de service est datree-webhook-server-awsmp.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *`my-cluster`* par le nom de votre cluster et *`my-datree-role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Datadog

Le nom du module complémentaire est `datadog_operator` et l'espace de noms est`datadog-agent`. Datadog publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation de l'agent Datadog sur Amazon EKS avec le module complémentaire Datadog Operator dans la documentation de Datadog.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Couvre-sol

Le nom du module complémentaire est `groundcover_agent` et l'espace de noms est`groundcover`. groundcover publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du module complémentaire Groundcover Amazon EKS dans la documentation Groundcover.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Laboratoires Grafana

Le nom du module complémentaire est `grafana-labs_kubernetes-monitoring` et l'espace de noms est`monitoring`. Grafana Labs publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Configurer Kubernetes Monitoring en tant que module complémentaire avec Amazon EKS dans la documentation de Grafana Labs.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Guance

- Editeur — GUANCE
- Nom – `guance_datakit`
- Espace de noms – `datakit`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- AWS stratégie IAM gérée : aucune stratégie gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.
- Instructions de configuration et d'utilisation — Consultez la section Utilisation du module complémentaire Amazon EKS dans la documentation de Guance.

Proxy HA

Le nom est `haproxy-technologies_kubernetes-ingress-ee` et l'espace de noms est`haproxy-controller`. HA Proxy publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Amazon EKS Integration dans la documentation de Datree.

Nom du compte de service

Le nom du compte de service est`customer defined`.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *my-cluster* par le nom de votre cluster et *my-haproxy-role* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kpow

Le nom du module complémentaire est `factorhouse_kpow` et l'espace de noms est`factorhouse`. Factorhouse publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez AWS Marketplace LM dans la documentation Kpow.

Nom du compte de service

Le nom du compte de service est`kpow`.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *my-cluster* par le nom de votre cluster et *my-kpow-role* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit

installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kubecost

Le nom du module complémentaire est `kubecost_kubecost` et l'espace de noms est`kubecost`. Kubecost publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Intégration de la facturation dans le AWS cloud dans la documentation de Kubecost.

Les volumes Store Kubernetes avec Amazon EBS doivent être installés sur votre cluster, sinon vous recevrez un message d'erreur.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kasten

Le nom du module complémentaire est `kasten_k10` et l'espace de noms est`kasten-io`. Kasten by Veeam publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du K10 à l' AWS aide du module complémentaire Amazon EKS dans la documentation de Kasten.

Le pilote Amazon EBS CSI doit être installé sur votre cluster avec un pilote par défaut. `StorageClass`

Nom du compte de service

Le nom du compte de service est`k10-k10`.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *`my-cluster`* par le nom de votre cluster et *`my-kasten-role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez. the section called "Attribuer un rôle IAM"

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kong

Le nom du module complémentaire est `kong_konnect-ri` et l'espace de noms est`kong`. Kong publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du module complémentaire Kong Gateway EKS dans la documentation de Kong.

Les volumes Store Kubernetes avec Amazon EBS doivent être installés sur votre cluster, sinon vous recevrez un message d'erreur.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

LeakSignal

Le nom du module complémentaire est `leaksignal_leakagent` et l'espace de noms est`leakagent`. LeakSignal publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez https://www.leaksignal.com/docs/LeakAgent/Deployment/ AWS%20Eks%20Addon/ [Installer le module complémentaire] dans la documentation LeakAgent LeakSignal

Les volumes Store Kubernetes avec Amazon EBS doivent être installés sur votre cluster, sinon vous recevrez un message d'erreur.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

NetApp

Le nom du module complémentaire est `netapp_trident-operator` et l'espace de noms est`trident`. NetApp publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, voir Configurer le module complémentaire Trident EKS dans la NetApp documentation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

New Relic

Le nom du module complémentaire est `new-relic_kubernetes-operator` et l'espace de noms est`newrelic`. New Relic publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du module complémentaire New Relic pour EKS dans la documentation de New Relic.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Rafay

Le nom du module complémentaire est `rafay-systems_rafay-operator` et l'espace de noms est`rafay-system`. Rafay publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du module complémentaire Rafay Amazon EKS dans la documentation de Rafay.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Sécurité routière

- Editeur — RAD SECURITY
- Nom – `rad-security_rad-security`
- Espace de noms – `ksoc`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- AWS stratégie IAM gérée : aucune stratégie gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.
- Instructions de configuration et d'utilisation — Voir Installation de Rad Through The AWS Marketplace dans la documentation de Rad Security.

SolarWinds

- Editeur — SOLARWINDS
- Nom – `solarwinds_swo-k8s-collector-addon`
- Espace de noms – `solarwinds`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- AWS stratégie IAM gérée : aucune stratégie gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.
- Instructions de configuration et d'utilisation : consultez la section Surveiller un cluster Amazon EKS dans la SolarWinds documentation.

En solo

Le nom du module complémentaire est `solo-io_istio-distro` et l'espace de noms est`istio-system`. Solo publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation d'Istio dans la documentation de Solo.io.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Snyk

- Editeur — SNYK
- Nom – `snyk_runtime-sensor`
- Espace de noms – `snyk_runtime-sensor`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- AWS stratégie IAM gérée : aucune stratégie gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.
- Instructions de configuration et d'utilisation — Voir le capteur d'exécution Snyk dans la documentation utilisateur de Snyk.

Stormforge

Le nom du module complémentaire est `stormforge_optimize-Live` et l'espace de noms est`stormforge-system`. Stormforge publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation de l' StormForge agent dans la StormForge documentation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Splunk

Le nom du module complémentaire est `splunk_splunk-otel-collector-chart` et l'espace de noms est`splunk-monitoring`. Splunk publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Installer le module complémentaire Splunk pour Amazon EKS dans la documentation Splunk.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Teleport

Le nom du module complémentaire est `teleport_teleport` et l'espace de noms est`teleport`. Teleport publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Fonctionnement de la téléportation dans la documentation relative à la téléportation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Tétrate

Le nom du module complémentaire est `tetrate-io_istio-distro` et l'espace de noms est`istio-system`. Tetrate Io publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez le site Web de Tetrate Istio Distro.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Crossplane universel Upbound

Le nom du module complémentaire est `upbound_universal-crossplane` et l'espace de noms est`upbound-system`. Upbound publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Upbound Universal Crossplane (UXP) dans la documentation Upbound.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Sous le vent

Le nom du module complémentaire est `upwind` et l'espace de noms est`upwind`. Upwind publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la documentation Upwind.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Création d'un module complémentaire Amazon EKS

Les modules complémentaires Amazon EKS sont des logiciels complémentaires pour les clusters Amazon EKS. Tous les modules complémentaires Amazon EKS :

- Incluez les derniers correctifs de sécurité et corrections de bogues.
- Sont validés par AWS pour fonctionner avec Amazon EKS.
- Réduisez la quantité de travail requise pour gérer le logiciel complémentaire.

Vous pouvez créer un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI. Si le module complémentaire nécessite un rôle IAM, consultez les détails du module complémentaire spécifique dans les modules complémentaires Amazon EKS pour en savoir plus sur la création du rôle.

Prérequis

Avant de créer un module complémentaire, procédez comme suit :

- Le cluster doit exister avant de créer un module complémentaire pour celui-ci. Pour de plus amples informations, veuillez consulter the section called “Créer un cluster ”.

- Vérifiez si votre module complémentaire nécessite un rôle IAM. Pour de plus amples informations, veuillez consulter the section called "Vérifiez la compatibilité".
- Vérifiez que la version du module complémentaire Amazon EKS est compatible avec votre cluster. Pour de plus amples informations, veuillez consulter the section called "Vérifiez la compatibilité".
- Vérifiez que la version 0.190.0 ou ultérieure de l'outil de ligne de `eksctl` commande est installée sur votre ordinateur ou. AWS CloudShell Pour plus d'informations, consultez Installation sur le site Web `eksctl`.

Procédure

Vous pouvez créer un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI. Si le module complémentaire nécessite un rôle IAM, consultez les informations relatives au module complémentaire spécifique dans la section Extensions Amazon EKS disponibles sur le site AWS pour en savoir plus sur la création du rôle.

Créer un module complémentaire (eksctl)

1. Affichez les noms des modules complémentaires disponibles pour une version de cluster. Remplacez *`1.33`* par la version de votre cluster.

   ```
   eksctl utils describe-addon-versions --kubernetes-version 1.33 | grep AddonName
   ```

   L'exemple qui suit illustre un résultat.

   ```
   "AddonName": "aws-ebs-csi-driver",
                           "AddonName": "coredns",
                           "AddonName": "kube-proxy",
                           "AddonName": "vpc-cni",
                           "AddonName": "adot",
                           "AddonName": "dynatrace_dynatrace-operator",
                           "AddonName": "upbound_universal-crossplane",
                           "AddonName": "teleport_teleport",
                           "AddonName": "factorhouse_kpow",
                           [...]
   ```

2. Consultez les versions disponibles pour le module complémentaire que vous souhaitez créer. Remplacez *`1.33`* par la version de votre cluster. Remplacez *`name-of-addon`* par le nom du

module complémentaire dont vous souhaitez afficher les versions. Le nom doit être l'un des noms renvoyés à l'étape précédente.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon |
 grep AddonVersion
```

La sortie suivante est un exemple de ce qui est renvoyé pour le module complémentaire nommé `vpc-cni`. Vous pouvez voir que le module complémentaire dispose de plusieurs versions disponibles.

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

a. Déterminez si le module complémentaire que vous souhaitez créer est un module complémentaire Amazon EKS ou AWS Marketplace. The AWS Marketplace propose des modules complémentaires tiers qui vous obligent à effectuer des étapes supplémentaires pour créer le module complémentaire.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-
addon | grep ProductUrl
```

Si aucune sortie n'est renvoyée, le module complémentaire est un module Amazon EKS. Si une sortie est renvoyée, le module complémentaire est un module complémentaire AWS Marketplace. La sortie suivante concerne un module complémentaire nommé `teleport_teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-
f96faef18b26"
```

Vous pouvez en savoir plus sur le module complémentaire sur le AWS Marketplace avec l'URL renvoyée. Si le module complémentaire nécessite un abonnement, vous pouvez vous y abonner via le AWS Marketplace. Si vous souhaitez créer un module complémentaire depuis AWS Marketplace, le principal IAM que vous utilisez pour créer le module complémentaire doit être autorisé à créer le rôle lié à un AWSServiceRoleForAWSLicenseManagerRoleservice. Pour plus

d'informations sur l'attribution d'autorisations à une entité IAM, consultez la rubrique Ajout et suppression d'autorisations basées sur l'identité IAM dans le Guide de l'utilisateur IAM.

3. Créez un module complémentaire Amazon EKS. Copiez la commande et remplacez-la *user-data* comme suit :

   - Remplacez *my-cluster* par le nom de votre cluster.
   - Remplacez *name-of-addon* par le nom du module complémentaire que vous souhaitez créer.
   - Si vous souhaitez une version du module complémentaire antérieure à la dernière version, remplacez-la *latest* par le numéro de version renvoyé dans le résultat de l'étape précédente que vous souhaitez utiliser.
   - Si le module complémentaire utilise un rôle de compte de service, remplacez *111122223333* par votre ID de compte et *role-name* par le nom du rôle. Pour obtenir des instructions sur la création d'un rôle pour votre compte de service, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

     Si le module complémentaire n'utilise aucun rôle de compte de service, supprimez-le`--service-account-role-arn arn:aws: iam::111122223333:role/role-name`.
   - Cet exemple de commande remplace la configuration de toute version autogérée existante du module complémentaire, le cas échéant. Si vous ne souhaitez pas remplacer la configuration d'un module complémentaire autogéré existant, supprimez l'*--force*option. Si vous supprimez cette option et que le module complémentaire doit remplacer la configuration d'un module complémentaire autogéré existant, la création du module complémentaire Amazon EKS échoue et vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option.

     ```
     eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
         --service-account-role-arn arn:aws: iam::111122223333:role/role-name --force
     ```

     Vous pouvez consulter la liste complète des options disponibles pour la commande.

     ```
     eksctl create addon --help
     ```

Pour plus d'informations sur les options disponibles, consultez Modules complémentaires dans la documentation `eksctl`.

Créer un module complémentaire (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez créer le module complémentaire.
4. Choisissez l'onglet Modules complémentaires.
5. Choisissez Obtenez plus de modules complémentaires.
6. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster. Vous pouvez ajouter autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez.

   Pour les modules complémentaires AWS Marketplace, le principal IAM que vous utilisez pour créer le module complémentaire doit être autorisé à lire les droits relatifs au module complémentaire depuis le. AWS LicenseManager AWS LicenseManager nécessite un rôle AWSServiceRoleForAWSLicenseManagerRolelié au service (SLR) qui permet aux AWS ressources de gérer les licences en votre nom. Le SLR est une exigence unique, par compte, et vous n'aurez pas à créer de SLR distinct pour chaque module complémentaire ni pour chaque cluster. Pour plus d'informations sur l'attribution d'autorisations à un principal IAM, consultez la rubrique Ajout et suppression d'autorisations basées sur l'identité IAM du Guide de l'utilisateur IAM.

   Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez cliquer sur la numérotation des pages pour afficher des résultats supplémentaires ou effectuer une recherche dans le champ de recherche. Dans les Options de filtrage, vous pouvez également filtrer par catégorie, fournisseur, ou modèle de tarification puis choisir les modules complémentaires dans les résultats de la recherche. Une fois que vous avez sélectionné les modules complémentaires que vous souhaitez installer, choisissez Next.
7. Sur la page Configurer les paramètres des modules complémentaires sélectionnés, procédez comme suit :

a. Choisissez Afficher les options d'abonnement pour ouvrir le formulaire des Options d'abonnement. Passez en revue les sections Détails de la tarification et Mentions légales, puis cliquez sur le bouton S'abonner pour continuer.

b. Dans Version, choisissez la version que vous souhaitez installer. Nous vous recommandons de choisir la version la plus récente, sauf si le module complémentaire que vous créez recommande une version différente. Pour déterminer si une version d'un module complémentaire est recommandée, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires".

c. Vous disposez de deux options pour configurer les rôles pour les modules complémentaires : le rôle IAM d'EKS Pod Identities et les rôles IAM pour les comptes de service (IRSA). Suivez l'étape appropriée ci-dessous pour l'option que vous préférez. Si tous les modules complémentaires que vous avez sélectionnés ont un abonnement sous État, cliquez sur Suivant. Vous ne pouvez pas configurer davantage ces modules complémentaires tant que vous ne vous y êtes pas abonné après la création de votre cluster. Pour les modules complémentaires dont le statut ne nécessite pas d'abonnement, procédez comme suit :

   i. Pour le rôle Pod Identity IAM pour le compte de service, vous pouvez soit utiliser un rôle EKS Pod Identity IAM existant, soit en créer un à l'aide du bouton Créer un rôle recommandé. Ce champ ne fournira que les options associées à la politique de confiance appropriée. S'il n'y a aucun rôle à sélectionner, cela signifie que vous n'avez aucun rôle existant avec une politique de confiance correspondante. Pour configurer un rôle IAM EKS Pod Identity pour les comptes de service du module complémentaire sélectionné, choisissez Create recommended role. L'assistant de création de rôles s'ouvre dans une fenêtre séparée. L'assistant renseigne automatiquement les informations relatives au rôle comme suit. Pour chaque module complémentaire dans lequel vous souhaitez créer le rôle EKS Pod Identity IAM, procédez comme suit dans l'assistant IAM.

      - À l'étape Sélectionner une entité de confiance, l'option de AWS service pour EKS et le cas d'utilisation pour EKS - Pod Identity sont présélectionnés, et la politique de confiance appropriée sera automatiquement renseignée pour le module complémentaire. Par exemple, le rôle sera créé avec la politique de confiance appropriée contenant le principal IAM pods.eks.amazonaws.com, comme indiqué dans. the section called "Avantages des identités du pod EKS" Choisissez Suivant.

      - À l'étape Ajouter des autorisations, la stratégie gérée appropriée pour la politique de rôle est présélectionnée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la politique gérée détaillée

`AmazonEKS_CNI_Policy` dans. the section called "Plugin CNI Amazon VPC pour Kubernetes" Choisissez Suivant.

- À l'étape Nom, révision et création, dans Nom du rôle, le nom du rôle par défaut est automatiquement renseigné pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé sous le nom Amazon. EKSPod IdentityAmazon VPCCNIRole Dans Description, la description par défaut est automatiquement renseignée avec la description appropriée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la description Autorise les pods exécutés dans le cluster Amazon EKS à accéder aux ressources. AWS Dans Politique de confiance, consultez la politique de confiance renseignée pour le module complémentaire. Choisissez Créer un rôle.

  REMARQUE : Le fait de conserver le nom de rôle par défaut permet à EKS de présélectionner le rôle pour les modules complémentaires dans les nouveaux clusters ou lors de l'ajout de modules complémentaires à des clusters existants. Vous pouvez toujours remplacer ce nom et le rôle sera disponible pour le module complémentaire dans tous vos clusters, mais le rôle devra être sélectionné manuellement dans le menu déroulant.

ii. Pour les modules complémentaires dont le statut ne nécessite pas d'abonnement et pour lesquels vous souhaitez configurer des rôles à l'aide d'IRSA, consultez la documentation du module complémentaire que vous créez pour créer une politique IAM et l'associer à un rôle. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". Pour sélectionner un rôle IAM, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

iii. Sélectionnez Optional configuration settings (Paramètres de configuration facultatifs).

iv. Si le module complémentaire nécessite une configuration, saisissez-la dans la zone Configuration values (Valeurs de configuration). Pour déterminer si le module complémentaire nécessite des informations de configuration, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires".

v. Choisissez l'une des options disponibles pour la méthode de résolution des conflits. Si vous choisissez Override pour la méthode de résolution des conflits, un ou plusieurs paramètres du module complémentaire existant peuvent être remplacés par les paramètres du module complémentaire Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur qui en résulte pour résoudre le conflit. Avant de choisir cette option, assurez-vous que le module

complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer vous-même.

vi. Choisissez Suivant.

8. Sur la page Vérifier et ajouter, choisissez Créer. Une fois l'installation des modules complémentaire terminée, vous pouvez voir les modules complémentaires installés.
9. Si l'un des modules complémentaires que vous avez installés nécessite un abonnement, procédez comme suit :
   a. Cliquez sur le bouton Subscribe (S'abonner) dans le coin inférieur droit du module complémentaire. Vous êtes redirigé vers la page du module complémentaire AWS sur le Marketplace. Lisez les informations relatives au module complémentaire, telles que Product Overview (Présentation du produit) et Pricing Information (Informations sur la tarification).
   b. Cliquez sur le bouton Continue to Subscribe (Continuer pour s'abonner) en haut à droite de la page du module complémentaire.
   c. Lisez les Conditions générales. Si vous les acceptez, sélectionnez Accept Terms (Accepter les conditions). Le traitement de l'abonnement peut prendre plusieurs minutes. Pendant le traitement de l'abonnement, le bouton Return to Amazon EKS Console (Retourner à la console Amazon EKS) est grisé.
   d. Une fois le traitement de l'abonnement terminé, le bouton Return to Amazon EKS Console (Retourner à la console Amazon EKS) n'est plus grisé. Cliquez sur le bouton pour revenir à l'onglet Add-ons (Modules complémentaires) de la console Amazon EKS pour votre cluster.
   e. Pour le module complémentaire auquel vous êtes abonné, sélectionnez Remove and reinstall (Supprimer et réinstaller), puis Reinstall add-on (Réinstaller le module complémentaire). L'installation du module complémentaire peut prendre plusieurs minutes. Une fois l'installation terminée, vous pouvez configurer le module complémentaire.

Créer un module complémentaire (AWS CLI)

1. Vous avez besoin d'`2.12.3`une version ou d'une version ultérieure `1.27.160` ou d'une version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport

à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.

2. Déterminez quels modules complémentaires sont disponibles. Vous pouvez voir tous les modules complémentaires disponibles, leur type et leur éditeur. Vous pouvez également consulter l'URL des modules complémentaires disponibles AWS sur le Marketplace. Remplacez *1.33* par la version de votre cluster.

```
aws eks describe-addon-versions --kubernetes-version 1.33 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
  |
+-----------------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
|                     MarketplaceProductUrl                     |             Name
           |      Owner       |  Publisher   |        Type         |
+-----------------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
|  None                                                         |  aws-ebs-csi-driver
           |  aws             |  eks         |  storage            |
|  None                                                         |  coredns
           |  aws             |  eks         |  networking         |
|  None                                                         |  kube-proxy
           |  aws             |  eks         |  networking         |
|  None                                                         |  vpc-cni
           |  aws             |  eks         |  networking         |
|  None                                                         |  adot
           |  aws             |  eks         |  observability      |
| https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
 |
| https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |  upbound_universal-
crossplane |  aws-marketplace |  upbound     |  infra-management   |
```

```
| https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |  teleport_teleport
          |  aws-marketplace |  teleport    |  policy-management  |
| https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |  factorhouse_kpow
          |  aws-marketplace |  factorhouse |  monitoring         |
|  [...]                                                        |  [...]
           |  [...]           |  [...]       |  [...]              |
+-----------------------------------------------------------------
+------------------------------+------------------+--------------
+---------------------+
```

Votre sortie peut être différente. Dans cet exemple de sortie, trois modules complémentaires différents sont disponibles par type `networking` et cinq modules complémentaires avec un éditeur de type `eks`. Les modules complémentaires figurant `aws-marketplace` dans la `Owner` colonne peuvent nécessiter un abonnement avant de pouvoir les installer. Vous pouvez consulter l'URL pour en savoir plus sur le module complémentaire et vous y abonner.

3. Vous pouvez voir quelles versions sont disponibles pour chaque module complémentaire. Remplacez *`1.33`* par la version de votre cluster et remplacez *`vpc-cni`* par le nom d'un module complémentaire renvoyé à l'étape précédente.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

La version avec `True` dans la colonne `Defaultversion` correspond à la version avec laquelle le module complémentaire est créé, par défaut.

4. (Facultatif) Recherchez les options de configuration du module complémentaire de votre choix en exécutant la commande suivante :

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.12.0-
eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema\":\"http://
json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":{\"additionalProperties
\":false,\"properties\":{\"hostPath\":{\"$ref\":\"#/definitions/HostPath\"}},\"title
\":\"Cri\",\"type\":\"object\"},\"Env\":{\"additionalProperties\":false,\"properties
\":{\"ADDITIONAL_ENI_TAGS\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":
\"integer\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL\":{\"type\":\"string\"},
\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL
\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION\":{\"format\":\"boolean\",\"type
\":\"string\"},\"DISABLE_METRICS\":{\"format\":\"boolean\",\"type\":\"string
\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING\":{\"format\":\"boolean\",\"type
\":\"string\"},\"ENABLE_POD_ENI\":{\"format\":\"boolean\",\"type\":\"string
\"},\"ENABLE_PREFIX_DELEGATION\":{\"format\":\"boolean\",\"type\":\"string\"},
\"WARM_ENI_TARGET\":{\"format\":\"integer\",\"type\":\"string\"},\"WARM_PREFIX_TARGET
\":{\"format\":\"integer\",\"type\":\"string\"}},\"title\":\"Env\",\"type\":
\"object\"},\"HostPath\":{\"additionalProperties\":false,\"properties\":{\"path
\":{\"type\":\"string\"}},\"title\":\"HostPath\",\"type\":\"object\"},\"Limits
\":{\"additionalProperties\":false,\"properties\":{\"cpu\":{\"type\":\"string
\"},\"memory\":{\"type\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},
\"Resources\":{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":
\"#/definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},\"title
\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties\":false,
\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":{\"$ref\":\"#/
definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/Resources\"}},\"title\":
\"VpcCni\",\"type\":\"object\"}}}"
}
```

La sortie correspond à un schéma JSON standard.

Voici un exemple de valeurs de configuration valides, au format JSON, fonctionnant avec le schéma ci-dessus.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

Voici un exemple de valeurs de configuration valides, au format YAML, fonctionnant avec le schéma ci-dessus.

```
  resources:
    limits:
      cpu: 100m
```

5. Déterminez si le module complémentaire nécessite des autorisations IAM. Dans ce cas, vous devez (1) déterminer si vous souhaitez utiliser EKS Pod Identities ou IAM Roles for Service Accounts (IRSA), (2) déterminer l'ARN du rôle IAM à utiliser avec le module complémentaire et (3) déterminer le nom du compte de service Kubernetes utilisé par le module complémentaire. Pour de plus amples informations, veuillez consulter the section called "Récupérez les informations IAM".
   - Amazon EKS suggère d'utiliser EKS Pod Identities si le module complémentaire le prend en charge. Cela nécessite que l'agent Pod Identity soit installé sur votre cluster. Pour plus d'informations sur l'utilisation de Pod Identities avec des modules complémentaires, consultezthe section called "Rôles IAM".
   - Si le module complémentaire ou votre cluster n'est pas configuré pour EKS Pod Identities, utilisez IRSA. Vérifiez que l'IRSA est configuré sur votre cluster.
   - Consultez la documentation relative aux modules complémentaires Amazon EKS pour déterminer si le module complémentaire nécessite des autorisations IAM et le nom du compte de service Kubernetes associé.
     a. Créez un module complémentaire Amazon EKS. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :
   - Remplacez `my-cluster` par le nom de votre cluster.

- Remplacez *vpc-cni* par un nom de module complémentaire renvoyé dans le résultat de l'étape précédente que vous souhaitez créer.
- Remplacez *version-number* par la version renvoyée dans le résultat de l'étape précédente que vous souhaitez utiliser.
- Si le module complémentaire ne nécessite pas d'autorisations IAM, *<service-account-configuration>* supprimez-le.
- Effectuez l'une des actions suivantes :
  - Si le module complémentaire (1) nécessite des autorisations IAM et (2) votre cluster utilise EKS Pod Identities, remplacez-le par *<service-account-configuration>* l'association d'identité de pod suivante. Remplacez *<service-account-name>* par le nom du compte de service utilisé par le module complémentaire. Remplacez *<role-arn>* par l'ARN d'un rôle IAM. Le rôle doit respecter la politique de confiance requise par EKS Pod Identities.

    ```
    --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
    ```

  - Si le module complémentaire (1) nécessite des autorisations IAM et (2) votre cluster utilise IRSA, remplacez-le *<service-account-configuration>* par la configuration IRSA suivante. *111122223333*Remplacez-le par votre identifiant de compte et *role-name* par le nom d'un rôle IAM existant que vous avez créé. Pour obtenir des instructions sur la création du rôle, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

    ```
    --service-account-role-arn arn:aws::iam::111122223333:role/role-name
    ```

- Cet exemple de commande remplace l'option `--configuration-values` de toute version autogérée existante du module complémentaire, le cas échéant. Remplacez cela par les valeurs de configuration souhaitées, telles qu'une chaîne ou une entrée de fichier. Si vous ne souhaitez pas fournir de valeurs de configuration, supprimez l'`--configuration-values`option. Si vous ne souhaitez pas que la AWS CLI remplace la configuration d'un module complémentaire autogéré existant, supprimez l'*--resolve-conflicts OVERWRITE*option. Si vous supprimez cette option et que le module complémentaire doit remplacer la configuration d'un module complémentaire autogéré existant, la création du module complémentaire Amazon EKS échoue

et vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
     <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    <service-account-configuration> --configuration-values 'file://example.yaml' --
resolve-conflicts OVERWRITE
```

Pour la liste complète des options disponibles, consultez `create-addon` dans la référence des lignes de commande Amazon EKS. Si le module complémentaire que vous avez créé `aws-marketplace` figure dans la `Owner` colonne d'une étape précédente, sa création peut échouer et vous pouvez recevoir un message d'erreur similaire au message d'erreur suivant.

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-on. To
 subscribe, visit the AWS Marketplace console, agree to the seller EULA, select the
 pricing type if required, then re-install the add-on"
                }
            ]
        }
    }
}
```

Si vous recevez une erreur similaire à celle de la sortie précédente, consultez l'URL figurant dans la sortie d'une étape précédente pour vous abonner au module complémentaire. Une fois abonné, exécutez à nouveau la commande `create-addon`.

Mettre à jour un module complémentaire Amazon EKS

Amazon EKS ne met pas automatiquement à jour un module complémentaire lorsque de nouvelles versions sont publiées ou après la mise à jour de votre cluster vers une nouvelle version mineure de Kubernetes. Pour mettre à jour un module complémentaire pour un cluster existant, vous devez lancer la mise à jour. Ensuite, Amazon EKS met à jour le module complémentaire Amazon EKS pour vous. Avant de mettre à jour un module complémentaire, consultez la documentation relative au module complémentaire. Pour obtenir la liste des modules complémentaires disponibles, consultez the section called " AWS modules complémentaires". Si le module complémentaire nécessite un rôle IAM, consultez les détails du module complémentaire spécifique dans la section Extensions Amazon EKS disponibles sur le site AWS pour en savoir plus sur la création du rôle.

Prérequis

Avant de créer un module complémentaire, procédez comme suit :

- Vérifiez si votre module complémentaire nécessite un rôle IAM. Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires Amazon EKS".
- Vérifiez que la version du module complémentaire Amazon EKS est compatible avec votre cluster. Pour de plus amples informations, veuillez consulter the section called "Vérifiez la compatibilité".

Procédure

Vous pouvez mettre à jour un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI.

Module de mise à jour (eksctl)

1. Déterminez les modules complémentaires et les versions de modules complémentaires actuellement installés sur votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

```
eksctl get addon --cluster my-cluster
```

L'exemple qui suit illustre un résultat.

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                v1.12.0-eksbuild.1,v1.11.4-
eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-eksbuild.1
```

Votre sortie peut être différente, en fonction des modules complémentaires et des versions dont vous disposez sur votre cluster. Vous pouvez voir que dans l'exemple de sortie précédent, deux modules complémentaires existants sur le cluster disposent de versions plus récentes disponibles dans la colonne `UPDATE AVAILABLE`.

2. Mettez à jour le module complémentaire.

   a. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande :

      - Remplacez *`my-cluster`* par le nom de votre cluster.
      - Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.
      - Remplacez *`vpc-cni`* par le nom d'un module complémentaire renvoyé dans la sortie de l'étape précédente que vous souhaitez mettre à jour.
      - Si vous souhaitez procéder à une mise à jour vers une version antérieure à la dernière version disponible, remplacez *`latest`* par le numéro de version renvoyé dans la sortie de l'étape précédente que vous souhaitez utiliser. Certains modules complémentaires présentent des versions recommandées. Pour plus d'informations, consultez la documentation du module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". * Si le module complémentaire utilise un compte de service Kubernetes et un rôle IAM, remplacez-les par *`111122223333`* votre identifiant de compte et *`role-name`* par le nom d'un rôle IAM existant que vous avez créé. Pour obtenir des instructions sur la création du rôle, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Si le module complémentaire n'utilise pas de compte de service Kubernetes ni de rôle IAM, supprimez la ligne. `serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name`

- L'*preserve*option préserve les valeurs existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres complémentaires et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `overwrite`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour des paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par`none`, Amazon EKS ne modifie pas la valeur des paramètres, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

b. Exécutez la commande modifiée pour créer le fichier `update-addon.yaml`.

c. Appliquez le fichier de configuration à votre cluster.

```
eksctl update addon -f update-addon.yaml
```

Pour plus d'informations sur la mise à jour des modules complémentaires, consultez la section Mise à jour des modules complémentaires dans la `eksctl` documentation.

Mettre à jour le module complémentaire (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez mettre à jour le module complémentaire.
4. Choisissez l'onglet Modules complémentaires.
5. Choisissez le module complémentaire que vous souhaitez mettre à jour.
6. Choisissez Modifier.
7. Sur la `name of addon` page Configurer, procédez comme suit :
   a. Choisissez la version que vous souhaitez utiliser. Le module complémentaire peut présenter une version recommandée. Pour plus d'informations, consultez la documentation du module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires".
   b. Vous disposez de deux options pour configurer les rôles pour les modules complémentaires : le rôle IAM d'EKS Pod Identities et les rôles IAM pour les comptes de service (IRSA). Suivez l'étape appropriée ci-dessous pour l'option que vous préférez. Si tous les modules complémentaires que vous avez sélectionnés ont un abonnement sous État, cliquez sur Suivant. Pour les modules complémentaires dont le statut ne nécessite pas d'abonnement, procédez comme suit :
      i. Pour le rôle Pod Identity IAM pour le compte de service, vous pouvez soit utiliser un rôle EKS Pod Identity IAM existant, soit en créer un à l'aide du bouton Créer un rôle recommandé. Ce champ ne fournira que les options associées à la politique de confiance appropriée. S'il n'y a aucun rôle à sélectionner, cela signifie que vous n'avez aucun rôle existant avec une politique de confiance correspondante. Pour configurer un rôle IAM EKS Pod Identity pour les comptes de service du module complémentaire sélectionné, choisissez Create recommended role. L'assistant de création de rôles s'ouvre dans une fenêtre séparée. L'assistant renseigne automatiquement les informations relatives au rôle comme suit. Pour chaque module complémentaire dans lequel vous souhaitez créer le rôle EKS Pod Identity IAM, procédez comme suit dans l'assistant IAM.
         - À l'étape Sélectionner une entité de confiance, l'option de AWS service pour EKS et le cas d'utilisation pour EKS - Pod Identity sont présélectionnés, et la politique de confiance appropriée sera automatiquement renseignée pour le module complémentaire. Par exemple, le rôle sera créé avec la politique de confiance appropriée contenant le principal IAM pods.eks.amazonaws.com, comme indiqué dans. the section called "Avantages des identités du pod EKS" Choisissez Suivant.

- À l'étape Ajouter des autorisations, la stratégie gérée appropriée pour la politique de rôle est présélectionnée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la politique gérée détaillée `AmazonEKS_CNI_Policy` dans. the section called "Plugin CNI Amazon VPC pour Kubernetes" Choisissez Suivant.
- À l'étape Nom, révision et création, dans Nom du rôle, le nom du rôle par défaut est automatiquement renseigné pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé sous le nom Amazon. EKSPod IdentityAmazon VPCCNIRole Dans Description, la description par défaut est automatiquement renseignée avec la description appropriée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la description Autorise les pods exécutés dans le cluster Amazon EKS à accéder aux ressources. AWS Dans Politique de confiance, consultez la politique de confiance renseignée pour le module complémentaire. Choisissez Créer un rôle.

> **Note**
>
> Le fait de conserver le nom de rôle par défaut permet à EKS de présélectionner le rôle pour les modules complémentaires dans les nouveaux clusters ou lors de l'ajout de modules complémentaires aux clusters existants. Vous pouvez toujours remplacer ce nom et le rôle sera disponible pour le module complémentaire dans tous vos clusters, mais le rôle devra être sélectionné manuellement dans le menu déroulant.

ii. Pour les modules complémentaires dont le statut ne nécessite pas d'abonnement et pour lesquels vous souhaitez configurer des rôles à l'aide d'IRSA, consultez la documentation du module complémentaire que vous créez pour créer une politique IAM et l'associer à un rôle. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". Pour sélectionner un rôle IAM, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

c. Sélectionnez Paramètres de configuration facultatifs.

d. Dans Valeurs de configuration, entrez les informations de configuration spécifiques au module complémentaire. Pour plus d'informations, consultez la documentation du module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires,

voirthe section called " AWS modules complémentaires"... Pour la méthode de résolution des conflits, sélectionnez l'une des options. Si vous avez défini des valeurs personnalisées pour les paramètres des modules complémentaires, nous vous recommandons l'option Preserve (Conserver). Si vous ne choisissez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous modifiez cette valeur pour remplacer, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour des paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par none, Amazon EKS ne modifie pas la valeur des paramètres, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.

8. Sélectionnez Enregistrer les modifications.

Module complémentaire de mise à jour (AWS CLI)

1. Vous avez besoin d'`2.12.3`une version ou d'une version ultérieure `1.27.160` ou d'une version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
2. Consultez la liste des modules complémentaires installés. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks list-addons --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
```

```
        "vpc-cni"
    ]
}
```

3. Affichez la version actuelle du module complémentaire que vous souhaitez mettre à jour. Remplacez *my-cluster* par le nom de votre cluster et *vpc-cni* par le nom du module complémentaire que vous souhaitez mettre à jour.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

L'exemple qui suit illustre un résultat.

```
v1.10.4-eksbuild.1
```

4. Déterminez quelles versions du module complémentaire sont disponibles pour la version de votre cluster. *1.33*Remplacez-le par la version de votre cluster et *vpc-cni* par le nom du module complémentaire que vous souhaitez mettre à jour.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

La version avec `True` dans la colonne `Defaultversion` correspond à la version avec laquelle le module complémentaire est créé, par défaut.

5. Mettez à jour votre module complémentaire. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande

modifiée. Pour plus d'informations sur cette commande, consultez update-addon dans le manuel Amazon EKS Command Line Reference.

- Remplacez *my-cluster* par le nom de votre cluster.
- Remplacez *vpc-cni* par le nom du module complémentaire que vous souhaitez mettre à jour renvoyé dans la sortie d'une étape précédente.
- Remplacez *version-number* par la version renvoyée dans la sortie de l'étape précédente vers laquelle vous souhaitez effectuer la mise à jour. Certains modules complémentaires présentent des versions recommandées. Pour plus d'informations, consultez la documentation du module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". * Si le module complémentaire utilise un compte de service Kubernetes et un rôle IAM, remplacez-les par *111122223333* votre identifiant de compte et *role-name* par le nom d'un rôle IAM existant que vous avez créé. Pour obtenir des instructions sur la création du rôle, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

  Si le module complémentaire n'utilise pas de compte de service Kubernetes ni de rôle IAM, supprimez la ligne. `serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name`
- L'`--resolve-conflicts PRESERVE`option préserve les valeurs existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres complémentaires et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `OVERWRITE`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour des paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par`NONE`, Amazon EKS ne modifie pas la valeur des paramètres, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.
- Si vous voulez supprimer toutes les configurations personnalisées, effectuez la mise à jour à l'aide de l'option *--configuration-values '{}'*. Cela ramène toutes les configurations personnalisées aux valeurs par défaut. Si vous ne souhaitez pas modifier votre configuration

personnalisée, ne fournissez pas le *--configuration-values* drapeau. Si vous voulez ajuster une configuration personnalisée, remplacez *{}* par les nouveaux paramètres.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

6. Vérifiez l'état de la mise à jour. *my-cluster*Remplacez-le par le nom de votre cluster et *vpc-cni* par le nom du module complémentaire que vous mettez à jour.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

L'exemple qui suit illustre un résultat.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
    }
}
```

Lorsque la mise à jour est terminé, l'état indique `ACTIVE`.

Vérifier la compatibilité de la version du module complémentaire Amazon EKS avec un cluster

Avant de créer un module complémentaire Amazon EKS, vous devez vérifier que la version du module complémentaire Amazon EKS est compatible avec votre cluster.

Utilisez l'describe-addon-versions API pour répertorier les versions disponibles des modules complémentaires EKS, ainsi que les versions de Kubernetes prises en charge par chaque version d'extension.

1. Vérifiez que la AWS CLI est installée et fonctionne avec`aws sts get-caller-identity`. Si cette commande ne fonctionne pas, découvrez comment démarrer avec la AWS CLI.
2. Déterminez le nom du module complémentaire pour lequel vous souhaitez récupérer les informations de compatibilité des versions, par exemple`amazon-cloudwatch-observability`.

3. Déterminez la version Kubernetes de votre cluster, par exemple. `1.33`
4. Utilisez la AWS CLI pour récupérer les versions des modules complémentaires compatibles avec la version Kubernetes de votre cluster.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.33
```

L'exemple qui suit illustre un résultat.

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "vX.X.X-eksbuild.X",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "computeTypes": [
                        "ec2",
                        "auto",
                        "hybrid"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.33",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                }
            ]
        }
    ]
}
```

Cette sortie indique que la version de l'addon `vX.X.X-eksbuild.X` est compatible avec la version du cluster Kubernetes. `1.33`

Compatibilité des modules complémentaires avec les types de calcul

Le `computeTypes` champ de `describe-addon-versions` sortie indique la compatibilité d'un module complémentaire avec les nœuds gérés en mode automatique ou les nœuds hybrides d'EKS. Les modules complémentaires marqués `auto` fonctionnent avec l'infrastructure AWS gérée basée sur le cloud d'EKS Auto Mode, tandis que ceux marqués `hybrid` peuvent être exécutés sur des nœuds locaux connectés au plan de contrôle cloud EKS.

Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique d'Amazon EKS".

Supprimer un module complémentaire Amazon EKS d'un cluster

Vous pouvez supprimer un module complémentaire Amazon EKS de votre cluster à l'aide `eksctl` de la AWS Management Console CLI ou de la AWS CLI.

Lorsque vous supprimez un module complémentaire Amazon EKS d'un cluster :

- Il n'y a pas de temps d'arrêt pour les fonctionnalités fournies par le module complémentaire.
- Si vous utilisez des rôles IAM pour les comptes de service (IRSA) et qu'un rôle IAM est associé au module complémentaire, celui-ci n'est pas supprimé.
- Si vous utilisez Pod Identities, toutes les associations d'identité de pod détenues par le module complémentaire sont supprimées. Si vous spécifiez l'`--preserve`option dans la AWS CLI, les associations sont préservées.
- Amazon EKS cesse de gérer les paramètres du module complémentaire.
- La console cesse de vous avertir lorsque de nouvelles versions sont disponibles.
- Vous ne pouvez pas mettre à jour le module complémentaire à l'aide d' AWS outils ou APIs.
- Vous pouvez choisir de laisser le module complémentaire sur votre cluster afin de pouvoir le gérer automatiquement, ou le supprimer de votre cluster. Vous ne devez supprimer le module complémentaire de votre cluster que si aucune ressource sur votre cluster ne dépend de la fonctionnalité fournie par ce module complémentaire.

Prérequis

Avant de créer un module complémentaire, procédez comme suit :

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Mise en route.
- Vérifiez si votre module complémentaire nécessite un rôle IAM. Pour plus d'informations, veuillez consulter la rubrique
- Version `0.210.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour effectuer une installation ou une mise à jour`eksctl`, consultez la section Installation dans la `eksctl` documentation.

Procédure

Deux options s'offrent à vous lorsque vous supprimez un module complémentaire Amazon EKS.

- Conserver le logiciel complémentaire sur votre cluster : cette option supprime la gestion de tous les paramètres par Amazon EKS. Elle supprime également la possibilité pour Amazon EKS de vous informer des mises à jour et de mettre automatiquement à jour le module complémentaire Amazon EKS après le lancement d'une mise à jour. Toutefois, elle préserve le logiciel complémentaire sur votre cluster. Cette option fait du module complémentaire une installation autogérée plutôt qu'un module complémentaire Amazon EKS. Avec cette option, il n'y a aucun temps d'arrêt pour le module complémentaire.
- Supprimer entièrement le logiciel complémentaire de votre cluster : nous vous recommandons de supprimer le module complémentaire Amazon EKS de votre cluster uniquement si aucune ressource de votre cluster n'en dépend.

Vous pouvez supprimer un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI.

Supprimer le module complémentaire (eksctl)

1. Déterminez les modules complémentaires actuellement installés sur votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   L'exemple qui suit illustre un résultat.

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
[...]
```

Votre sortie peut être différente, en fonction des modules complémentaires et des versions dont vous disposez sur votre cluster.

2. Supprimez le module complémentaire. Remplacez *my-cluster* par le nom de votre cluster et *name-of-add-on* par le nom du module complémentaire renvoyé dans la sortie de l'étape précédente que vous souhaitez supprimer. Si vous supprimez *--preserve* cette option, en plus du fait qu'Amazon EKS ne gère plus le module complémentaire, le logiciel complémentaire est supprimé de votre cluster.

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

Pour plus d'informations sur la suppression des modules complémentaires, consultez la section Suppression des modules complémentaires dans la `eksctl` documentation.

Supprimer le module complémentaire (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez supprimer le module complémentaire Amazon EKS.
4. Choisissez l'onglet Modules complémentaires.
5. Choisissez le module complémentaire que vous souhaitez supprimer.
6. Sélectionnez Remove (Supprimer).
7. Dans la boîte de dialogue Supprimer : ***name of addon*** confirmation, procédez comme suit :
   a. Si vous voulez qu'Amazon EKS cesse de gérer les paramètres du module complémentaire, sélectionnez Conserver sur le cluster. Faites ceci si vous voulez retenir le logiciel du module complémentaire sur votre cluster. Ceci afin que vous puissiez gérer vous-même tous les paramètres du module complémentaire.
   b. Entrez le nom du module complémentaire.

c. Sélectionnez Remove (Supprimer).

Supprimer le module complémentaire (AWS CLI)

1. Vous avez besoin d'`0.210.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Consultez la liste des modules complémentaires installés. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks list-addons --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

3. Supprimez le module complémentaire installé. Remplacez *`my-cluster`* par le nom de votre cluster et *`name-of-add-on`* par le nom du module complémentaire que vous souhaitez supprimer. La suppression *`--preserve`* supprime le logiciel complémentaire de votre cluster.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
```

L'exemple suivant illustre le résultat abrégé.

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
    }
}
```

4. Vérifiez l'état de la suppression. *my-cluster*Remplacez-le par le nom de votre cluster et *name-of-addon* par le nom du module complémentaire que vous supprimez.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

Une fois le module complémentaire supprimé, l'exemple de sortie est le suivant.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

Rôles IAM pour les modules complémentaires Amazon EKS

Certains modules complémentaires Amazon EKS nécessitent des rôles et des autorisations IAM pour appeler AWS APIs. Par exemple, le module complémentaire Amazon VPC CNI appelle certains AWS APIs pour configurer les ressources réseau de votre compte. Ces modules complémentaires doivent être autorisés à l'aide d'IAM. Plus précisément, le compte de service du pod exécutant le module complémentaire doit être associé à un rôle IAM avec une politique IAM spécifique.

La méthode recommandée pour accorder des AWS autorisations aux charges de travail de cluster est d'utiliser la fonctionnalité Pod Identities d'Amazon EKS. Vous pouvez utiliser une association d'identité Pod pour associer le compte de service d'un module complémentaire à un rôle IAM. Si un pod utilise un compte de service qui a une association, Amazon EKS définit des variables d'environnement dans les conteneurs du pod. Les variables d'environnement configurent AWS SDKs, y compris la AWS CLI, pour utiliser les informations d'identification EKS Pod Identity. Pour plus d'informations, consultez the section called "Identité du pod".

Les modules complémentaires Amazon EKS peuvent aider à gérer le cycle de vie des associations d'identité des pods correspondant au module complémentaire. Par exemple, vous pouvez créer ou mettre à jour un module complémentaire Amazon EKS et l'association d'identité de pod nécessaire en un seul appel d'API. Amazon EKS fournit également une API permettant de récupérer les politiques IAM suggérées.

1. Vérifiez que l'agent d'identité du pod Amazon EKS est configuré sur votre cluster.
2. Déterminez si le module complémentaire que vous souhaitez installer nécessite des autorisations IAM à l'aide de la `describe-addon-versions` AWS CLI. Si l'`requiresIamPermissions`indicateur l'est`true`, vous devez utiliser l'`describe-addon-`

`configurationsopération` pour déterminer les autorisations requises par l'addon. La réponse inclut une liste de stratégies IAM gérées suggérées.

3. Récupérez le nom du compte de service Kubernetes et la politique IAM à l'aide de l'opération CLI. `describe-addon-configuration` Évaluez la portée de la politique suggérée par rapport à vos exigences de sécurité.
4. Créez un rôle IAM en utilisant la politique d'autorisation suggérée et la politique de confiance requise par Pod Identity. Pour de plus amples informations, veuillez consulter the section called "Création d'une association Pod Identity (AWS console)".
5. Créez ou mettez à jour un module complémentaire Amazon EKS à l'aide de la CLI. Spécifiez au moins une association d'identité de pod. Une association d'identité de pod est le nom d'un compte de service Kubernetes et l'ARN du rôle IAM.
   - Les associations d'identité de pod créées à l'aide du module complémentaire APIs appartiennent au module complémentaire correspondant. Si vous supprimez le module complémentaire, l'association d'identité du module est également supprimée. Vous pouvez empêcher cette suppression en cascade en utilisant l'`preserve`option lors de la suppression d'un addon à l'aide de la AWS CLI ou de l'API. Vous pouvez également mettre à jour ou supprimer directement l'association d'identité du module si nécessaire. Les modules complémentaires ne peuvent pas assumer la propriété des associations d'identité de pods existantes. Vous devez supprimer l'association existante et la recréer à l'aide d'une opération de création ou de mise à jour d'un module complémentaire.
   - Amazon EKS recommande d'utiliser les associations d'identité des pods pour gérer les autorisations IAM pour les modules complémentaires. La méthode précédente, les rôles IAM pour les comptes de service (IRSA), est toujours prise en charge. Vous pouvez spécifier à la fois une association d'identité IRSA `serviceAccountRoleArn` et une association d'identité de pod pour un module complémentaire. Si l'agent d'identité du pod EKS est installé sur le cluster, il `serviceAccountRoleArn` sera ignoré et EKS utilisera l'association d'identité du pod fournie. Si Pod Identity n'est pas activé, il `serviceAccountRoleArn` sera utilisé.
   - Si vous mettez à jour les associations d'identité des modules pour un module complémentaire existant, Amazon EKS lance un redémarrage progressif des modules complémentaires.

Récupérez les informations IAM relatives à un module complémentaire Amazon EKS

Avant de créer un module complémentaire, utilisez la AWS CLI pour déterminer :

- Si le module complémentaire nécessite des autorisations IAM

- La politique IAM suggérée à utiliser

Procédure

1. Déterminez le nom du module complémentaire que vous souhaitez installer et la version Kubernetes de votre cluster. Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".
2. Utilisez la AWS CLI pour déterminer si le module complémentaire nécessite des autorisations IAM.

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
--kubernetes-version <kubernetes-version>
```

Par exemple :

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

Passez en revue l'exemple de sortie suivant. Notez que `requiresIamPermissions` c'est `true` la version du module complémentaire par défaut. Vous devez spécifier la version du module complémentaire lors de la récupération de la politique IAM recommandée.

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
```

```
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. Si le module complémentaire nécessite des autorisations IAM, utilisez la AWS CLI pour récupérer une politique IAM recommandée.

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

Par exemple :

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

Passez en revue le résultat suivant. Notez le `recommendedManagedPolicies`.

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
]
```

4. Créez un rôle IAM et associez la politique gérée recommandée. Vous pouvez également passer en revue la politique gérée et limiter les autorisations le cas échéant. Pour de plus amples informations, veuillez consulter the section called "Création d'une association Pod Identity (AWS console)".

Référence de support Pod Identity

Le tableau suivant indique si certains modules complémentaires Amazon EKS sont compatibles avec EKS Pod Identity.

Nom du module	Prise en charge de l'identité des	Version minimale requise
Pilote Amazon EBS CSI	Oui	v1.26.0-eksbuild.1
CNI Amazon VPC	Oui	v1.15.5-eksbuild.1
Pilote CSI Amazon EFS	Oui	v2.0.5-eksbuild.1
AWS Distro pour OpenTelemetry	Oui	v0.94.1-eksbuild.1
Pilote CSI Mountpoint pour Amazon S3	Non	N/A
Agent Amazon CloudWatch Observability	Oui	v3.1.0-eksbuild.1

Ce tableau a été mis à jour pour la dernière fois le 28 octobre 2024.

Utilisez Pod Identities pour attribuer un rôle IAM à un module complémentaire Amazon EKS

Certains modules complémentaires Amazon EKS nécessitent des rôles et des autorisations IAM. Avant d'ajouter une mise à jour à un module complémentaire Amazon EKS pour utiliser une association Pod Identity, vérifiez le rôle et la politique à utiliser. Pour de plus amples informations, veuillez consulter the section called “Récupérez les informations IAM”.

1. Déterminez :
   - `cluster-name`— Nom du cluster sur lequel installer le module complémentaire.
   - `addon-name`— Nom du module complémentaire à installer.
   - `service-account-name`— Le nom du compte de service Kubernetes utilisé par le module complémentaire.

- `iam-role-arn`— L'ARN d'un rôle IAM doté d'autorisations suffisantes pour le module complémentaire. Le rôle doit disposer de la politique de confiance requise pour EKS Pod Identity. Pour plus d'informations, voir the section called "Création d'une association Pod Identity (AWS console)".

2. Mettez à jour le module complémentaire à l'aide de la AWS CLI. Vous pouvez également spécifier les associations Pod Identity lors de la création d'un module complémentaire, en utilisant la même `--pod-identity-assocations` syntaxe. Notez que lorsque vous spécifiez des associations d'identité d'espace lors de la mise à jour d'un module complémentaire, toutes les associations d'identité d'espace précédentes sont remplacées.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-
arn>'
```

Par exemple :

```
aws eks update-addon --cluster-name mycluster \
--addon-name aws-ebs-csi-driver \
--pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:
 iam::123456789012:role/StorageDriver'
```

3. Vérifiez que l'association Pod Identity a été créée :

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

Si la commande aboutit, vous devriez obtenir une sortie similaire à ce qui suit. Notez le OwnerArn du module complémentaire EKS.

```
{
    "associations": [
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws: eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
```

```
            "ownerArn": "arn:aws: eks:us-west-2:123456789012:addon/mycluster/aws-ebs-
csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

Supprimer les associations Pod Identity d'un module complémentaire Amazon EKS

Supprimez les associations Pod Identity d'un module complémentaire Amazon EKS.

1. Déterminez :
   - `cluster-name`- Le nom du cluster EKS sur lequel installer le module complémentaire.
   - `addon-name`- Le nom du module complémentaire Amazon EKS à installer.
2. Mettez à jour l'addon pour spécifier un tableau vide d'associations d'identité de pod.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

Résolution des problèmes liés à Pod Identities pour les modules complémentaires EKS

Si vos modules complémentaires rencontrent des erreurs lors de leurs tentatives d' AWS utilisation de l'API, du SDK ou de la CLI, vérifiez les points suivants :

- L'agent Pod Identity est installé dans votre cluster.
  - Pour plus d'informations sur l'installation de l'agent Pod Identity, consultezthe section called "Configuration de l'agent".
- Le module complémentaire possède une association Pod Identity valide.
  - Utilisez la AWS CLI pour récupérer les associations associées au nom du compte de service utilisé par le module complémentaire.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- Le rôle IAM dispose de la politique de confiance requise pour Pod Identities.
  - Utilisez la AWS CLI pour récupérer la politique de confiance d'un module complémentaire.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- Le rôle IAM dispose des autorisations nécessaires pour le module complémentaire.
  - AWS CloudTrail À utiliser pour passer en revue `AccessDenied` des `UnauthorizedOperation` événements.
- Le nom du compte de service indiqué dans l'association d'identité du module correspond au nom du compte de service utilisé par le module complémentaire.
  - Pour plus d'informations sur les modules complémentaires disponibles, consultezthe section called " AWS modules complémentaires".
- Vérifiez la configuration de MutatingWebhookConfiguration named `pod-identity-webhook`
  - `admissionReviewVersions`du webhook doit l'être `v1beta1` et ne fonctionne pas avec`v1`.

Déterminez les champs que vous pouvez personnaliser pour les modules complémentaires Amazon EKS

Les modules complémentaires Amazon EKS sont installés sur votre cluster à l'aide des configurations standard de bonnes pratiques. Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Modules complémentaires Amazon EKS".

Vous voudrez peut-être personnaliser la configuration d'un module complémentaire Amazon EKS pour activer les fonctions avancées. Amazon EKS utilise la fonctionnalité d'application côté serveur de Kubernetes pour permettre à Amazon EKS de gérer un module complémentaire sans remplacer votre configuration par des paramètres qui ne sont pas gérés par Amazon EKS. Pour plus d'informations, consultez apply côté serveur dans la documentation Kubernetes. Pour ce faire, Amazon EKS gère un ensemble minimal de champs pour chaque module complémentaire qu'il installe. Vous pouvez modifier tous les champs qui ne sont pas gérés par Amazon EKS ou par un autre processus du plan de contrôle Kubernetes, par exemple`kube-controller-manager`, sans problème.

> **Important**
>
> La modification d'un champ géré par Amazon EKS empêche Amazon EKS de gérer le module complémentaire. Ce processus peut entraîner l'écrasement de vos modifications lors de la mise à jour d'un module complémentaire.

Syntaxe de gestion des champs

Lorsque vous affichez les détails d'un objet Kubernetes, les champs gérés et non gérés sont renvoyés en sortie. Les champs gérés peuvent être l'un des types suivants :

- Entièrement géré : toutes les clés du champ sont gérées par Amazon EKS. Les modifications apportées à une valeur provoquent un conflit.
- Partiellement géré : certaines clés du champ sont gérées par Amazon EKS. Seules les modifications apportées aux clés explicitement gérées par Amazon EKS provoquent un conflit.

Les deux types de champs sont étiquetés avec `manager: eks`.

Chaque clé est soit un `.` représentant le champ lui-même, qui correspond toujours à un ensemble vide, soit une chaîne qui représente un sous-champ ou un élément. La sortie pour la gestion des champs comprend les types de déclarations suivants :

- `f:`*`name`* , où *`name`* est le nom d'un champ d'une liste.
- `k:`*`keys`* , où se *`keys`* trouve une carte des champs d'un élément de liste.
- `v:`*`value`* , où *`value`* est la valeur exacte au format JSON formaté de l'élément d'une liste.
- `i:`*`index`* , où *`index`* est la position d'un élément dans la liste.

Les parties de sortie suivantes pour le module complémentaire CoreDNS illustrent les déclarations précédentes :

- Champs entièrement gérés : si un champ géré possède un (champ) `f:` spécifié, mais aucune (clé) `k:`, alors le champ entier est géré. Les modifications apportées aux valeurs de ce champ provoquent un conflit.

  Dans la sortie suivante, vous pouvez voir que le conteneur nommé `coredns` est géré par `eks`. Les sous-champs `args`, `image` et `imagePullPolicy` sont également gérés par `eks`. Les modifications apportées aux valeurs de ces champs provoquent un conflit.

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
  ```

```
  f:imagePullPolicy: {}
[...]
manager: eks
[...]
```

- Champs partiellement gérés : si une clé gérée a une valeur spécifiée, les clés déclarées sont gérées pour ce champ. La modification des clés spécifiées provoque un conflit.

  Dans la sortie suivante, vous pouvez voir que `eks` gère les volumes `config-volume` et `tmp` définis à l'aide de la clé `name`.

```
[...]
f:volumes:
  k:{"name":"config-volume"}:
    .: {}
    f:configMap:
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"tmp"}:
    .: {}
    f:name: {}
[...]
manager: eks
[...]
```

- Ajout de clés à des champs partiellement gérés : si seule une valeur de clé spécifique est gérée, vous pouvez ajouter en toute sécurité des clés supplémentaires, telles que des arguments, à un champ sans provoquer de conflit. Si vous ajoutez des clés supplémentaires, assurez-vous d'abord que le champ n'est pas géré. L'ajout ou la modification d'une valeur gérée provoque un conflit.

  Dans la sortie suivante, vous pouvez voir que la clé `name` et le champ `name` sont gérés. L'ajout ou la modification d'un nom de conteneur provoque un conflit avec cette clé gérée.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

Procédure

Vous pouvez utiliser `kubectl` pour voir quels champs sont gérés par Amazon EKS pour tout module complémentaire Amazon EKS.

Vous pouvez modifier tous les champs qui ne sont pas gérés par Amazon EKS ou par un autre processus du plan de contrôle Kubernetes, par exemple`kube-controller-manager`, sans problème.

1. Déterminez le module complémentaire que vous souhaitez examiner. Pour voir l'ensemble du `deployments` sable DaemonSets déployé sur votre cluster, consultezthe section called "Accès aux ressources du cluster".
2. Pour afficher les champs gérés d'un module complémentaire, exécutez la commande suivante :

   ```
   kubectl get type/add-on-name -n add-on-namespace -o yaml
   ```

   Par exemple, vous pouvez voir les champs gérés pour le module complémentaire CoreDNS avec la commande suivante.

   ```
   kubectl get deployment/coredns -n kube-system -o yaml
   ```

   La gestion des champs est répertoriée dans la section suivante dans la sortie renvoyée.

   ```
   [...]
   managedFields:
     - apiVersion: apps/v1
       fieldsType: FieldsV1
       fieldsV1:
   [...]
   ```

   > **Note**
   >
   > Si vous ne le voyez pas `managedFields` dans le résultat, ajoutez-le `--show-managed-fields` à la commande et exécutez-la à nouveau. La version `kubectl` que vous utilisez détermine si les champs gérés sont renvoyés par défaut.

Étapes suivantes

Personnalisez les champs qui ne sont pas détenus par AWS le module complémentaire for you.

Valider les signatures des images du conteneur lors du déploiement

Si vous utilisez AWS Signer et que vous souhaitez vérifier les images de conteneur signées au moment du déploiement, vous pouvez utiliser l'une des solutions suivantes :

- Gatekeeper et Ratify : utilisez Gatekeeper comme contrôleur d'admission et Ratify configuré avec un plugin AWS Signer comme webhook pour valider les signatures.
- Kyverno — Un moteur de politique Kubernetes configuré avec un plugin AWS Signer pour valider les signatures.

> **Note**
>
> Avant de vérifier les signatures des images des conteneurs, configurez le référentiel de confiance Notation et la politique de confiance, comme l'exige le contrôleur d'admission que vous avez sélectionné.

Organiser et surveiller les ressources du cluster

Ce chapitre porte sur les rubriques suivantes relatives à la gestion de votre cluster. Vous pouvez également consulter les informations relatives à vos ressources Kubernetes à l'aide du. AWS Management Console

- Le tableau de bord Kubernetes est une interface utilisateur Web à usage général pour les clusters Kubernetes. Il permet aux utilisateurs de gérer les applications exécutées dans le cluster et de les dépanner, ainsi que de gérer le cluster lui-même. Pour plus d'informations, consultez le référentiel du tableau de bord Kubernetes. GitHub
- the section called "Serveur de métriques" : le serveur de métriques Kubernetes est un agrégateur de données d'utilisation des ressources dans votre cluster. Il n'est pas déployé par défaut dans votre cluster, mais est utilisé par les modules complémentaires Kubernetes, tels que le tableau de bord Kubernetes et. the section called "Horizontal Pod Autoscaler" Dans cette rubrique, vous allez apprendre à installer le serveur de métriques.
- the section called "Déployez des applications avec Helm" : le gestionnaire de paquets Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes. Cette rubrique vous aide à installer et à exécuter les binaires Helm, afin de pouvoir installer et gérer des graphiques à l'aide de la CLI Helm sur votre ordinateur local.
- the section called "Balisage de vos ressources " : pour vous aider à gérer vos ressources Amazon EKS, vous pouvez attribuer vos propres métadonnées à chaque ressource sous forme d'identifications. Cette rubrique décrit les identifications et explique comment les créer.
- the section called "Quotas de service"— Votre AWS compte dispose de quotas par défaut, anciennement appelés limites, pour chaque AWS service. Découvrez les quotas pour Amazon EKS et la manière de les augmenter.

Surveillez et optimisez les coûts du cluster Amazon EKS

La surveillance des coûts est un aspect essentiel de la gestion de vos clusters Kubernetes sur Amazon EKS. En obtenant une meilleure visibilité sur les coûts de votre cluster, vous pouvez optimiser l'utilisation des ressources, définir des budgets et prendre des décisions basées sur les données concernant vos déploiements. Amazon EKS propose deux solutions de surveillance des coûts, chacune présentant ses propres avantages uniques, pour vous aider à suivre et à répartir vos coûts de manière efficace :

AWS Données de répartition des coûts de facturation pour Amazon EKS : cette fonctionnalité native s'intègre parfaitement à la console de AWS facturation, vous permettant d'analyser et de répartir les coûts à l'aide de la même interface et des mêmes flux de travail que ceux que vous utilisez pour les autres AWS services. Grâce à la répartition des coûts, vous pouvez obtenir des informations sur vos coûts Kubernetes directement en parallèle avec vos autres AWS dépenses, ce qui facilite l'optimisation globale des coûts dans l'ensemble de votre environnement. AWS Vous pouvez également tirer parti des fonctionnalités AWS de facturation existantes, telles que Cost Categories et Cost Anomaly Detection, pour améliorer encore vos capacités de gestion des coûts. Pour plus d'informations, consultez la section Comprendre les données de répartition des coûts partagés dans le Guide AWS de l'utilisateur de facturation.

Kubecost — Amazon EKS prend en charge Kubecost, un outil de surveillance des coûts Kubernetes. Kubecost propose une approche riche en fonctionnalités et native de Kubernetes en matière de surveillance des coûts, fournissant des ventilations détaillées des coûts par ressources Kubernetes, des recommandations d'optimisation des coûts, ainsi que des tableaux de bord et des rapports. out-of-the-box Kubecost récupère également des données de tarification précises en les intégrant au rapport sur les AWS coûts et l'utilisation, ce qui vous permet d'obtenir une vue précise de vos coûts Amazon EKS. Découvrez comment installer Kubecost. Consultez la page Kubecost AWS Marketplace pour obtenir des informations sur l'obtention d'un abonnement gratuit à Kubecost.

Afficher les coûts par module dans la AWS facturation avec répartition des coûts fractionnée

Surveillance des coûts à l'aide des données de répartition des coûts AWS fractionnés pour Amazon EKS

Vous pouvez utiliser AWS les données de répartition des coûts pour Amazon EKS afin d'obtenir une visibilité précise des coûts pour vos clusters Amazon EKS. Cela vous permet d'analyser, d'optimiser et de rétrofacturer les coûts et l'utilisation de vos applications Kubernetes. Vous répartissez les coûts des applications entre les unités commerciales et les équipes individuelles en fonction des ressources de EC2 processeur et de mémoire Amazon consommées par votre application Kubernetes. Les données de répartition des coûts pour Amazon EKS donnent une visibilité sur le coût par pod et vous permettent d'agréger les données de coût par pod à l'aide de l'espace de noms, du cluster et d'autres primitives Kubernetes. Vous trouverez ci-dessous des exemples de primitives Kubernetes que vous pouvez utiliser pour analyser les données de répartition des coûts Amazon EKS.

- Nom du cluster

- Déploiement
- Espace de noms
- Nœud
- Nom de la charge de travail
- Type de charge de travail

Les balises de répartition des coûts définies par l'utilisateur sont également prises en charge. Pour plus d'informations sur l'utilisation des données de répartition des coûts fractionnés, consultez la section Comprendre les données de répartition des coûts dans le Guide AWS de l'utilisateur de facturation.

Configuration des rapports sur les coûts et l'utilisation

Vous pouvez activer les données de répartition des coûts pour EKS dans la console de gestion des coûts, l'interface de ligne de AWS commande ou le AWS SDKs.

Utilisez ce qui suit pour les données de répartition des coûts fractionnés :

1. Choisissez de fractionner les données de répartition des coûts. Pour plus d'informations, consultez la section Activation des données de répartition des coûts fractionnés dans le Guide de l'utilisateur du rapport sur les AWS coûts et l'utilisation.
2. Incluez les données dans un rapport nouveau ou existant.
3. Consultez le rapport. Vous pouvez utiliser la console de Facturation et gestion des coûts ou consulter les fichiers de rapports dans Amazon Simple Storage Service.

Installez Kubecost

Amazon EKS prend en charge Kubecost, que vous pouvez utiliser pour surveiller vos coûts ventilés par ressources Kubernetes, notamment les pods, les nœuds, les espaces de noms et les étiquettes. Cette rubrique traite de l'installation de Kubecost et de l'accès au tableau de bord Kubecost.

Amazon EKS fournit un ensemble AWS optimisé de Kubecost pour la visibilité des coûts des clusters. Vous pouvez utiliser vos contrats de AWS support existants pour obtenir de l'aide. Pour plus d'informations sur les versions disponibles de Kubecost, consultez. the section called “En savoir plus sur Kubecost”

> **Note**
>
> Kubecost v2 introduit plusieurs nouvelles fonctionnalités majeures. En savoir plus sur Kubecost v2.

Pour plus d'informations sur Kubecost, consultez la documentation de Kubecost et les questions fréquemment posées.

Installez le bundle Kubecost optimisé pour Amazon EKS

Vous pouvez utiliser l'une des procédures suivantes pour installer le bundle Kubecost optimisé pour Amazon EKS :

- Avant de commencer, il est recommandé de consulter Kubecost - Présentation de l'architecture pour comprendre le fonctionnement de Kubecost sur Amazon EKS.
- Si vous utilisez Amazon EKS pour la première fois, nous vous recommandons d'utiliser le module complémentaire Amazon EKS pour l'installation, car il simplifie l'installation du bundle Kubecost optimisé pour Amazon EKS. Pour plus d'informations, consultez Déploiement de Kubecost sur un cluster Amazon EKS à l'aide du module complémentaire Amazon EKS.
- Pour personnaliser l'installation, vous pouvez configurer votre bundle Kubecost optimisé pour Amazon EKS avec Helm. Pour plus d'informations, consultez la section Déploiement de Kubecost sur un cluster Amazon EKS à l'aide de Helm dans la documentation de Kubecost.

Accédez au tableau de bord Kubecost

Une fois la configuration du bundle Kubecost optimisé pour Amazon EKS terminée, vous devriez avoir accès au tableau de bord Kubecost. Pour de plus amples informations, veuillez consulter the section called “Accédez au tableau de bord Kubecost”.

Accédez au tableau de bord Kubecost

Prérequis

1. Assurez-vous que l'état des pods liés à Kubecost est « En cours d'exécution ».

```
kubectl get pods --namespace kubecost
```

Accédez au tableau de bord Kubecost

1. Sur votre appareil, activez le transfert de port pour exposer le tableau de bord Kubecost.

   ```
   kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
   ```

   Vous pouvez également utiliser le AWS Load Balancer Controller pour exposer Kubecost et utiliser Amazon Cognito pour l'authentification, les autorisations et la gestion des utilisateurs. Pour plus d'informations, consultez Comment utiliser Application Load Balancer et Amazon Cognito pour authentifier les utilisateurs de vos applications Web Kubernetes.

2. Depuis le même appareil qui vous a servi à effectuer l'étape précédente, ouvrez un navigateur Web et entrez l'adresse suivante.

   ```
   http://localhost:9090
   ```

   La page de présentation de Kubecost s'affiche dans votre navigateur. Selon la taille de votre cluster, Kubecost peut prendre 5 à 10 minutes (ou plus) pour recueillir des métriques. Vous pouvez consulter vos dépenses Amazon EKS, y compris les coûts cumulés du cluster, les coûts des actifs Kubernetes associés et les dépenses agrégées mensuelles.

3. Pour suivre les coûts au niveau du cluster, balisez vos ressources Amazon EKS pour la facturation. Pour de plus amples informations, veuillez consulter the section called "Identification de vos ressources pour facturation".

   - Allocation des coûts : Consultez les coûts mensuels d'Amazon EKS et les coûts cumulés pour chacun de vos espaces de noms et d'autres dimensions au cours des sept derniers jours. Cela est utile pour comprendre quels secteurs de votre application contribuent aux dépenses d'Amazon EKS.
   - Ressources : consultez les coûts des actifs AWS d'infrastructure associés à vos ressources Amazon EKS.

En savoir plus sur Kubecost

Amazon EKS fournit un ensemble AWS optimisé de Kubecost pour la visibilité des coûts des clusters. Amazon EKS prend en charge Kubecost, que vous pouvez utiliser pour surveiller vos coûts ventilés par ressources Kubernetes, notamment les pods, les nœuds, les espaces de noms et les étiquettes.

Cette rubrique couvre les versions disponibles de Kubecost et les différences entre les niveaux disponibles. EKS prend en charge les versions 1 et 2 de Kubecost. Chaque version est disponible en différents niveaux. Vous pouvez utiliser le bundle Kubecost optimisé pour Amazon EKS pour vos clusters Amazon EKS sans frais supplémentaires. L'utilisation de AWS services associés, tels qu'Amazon Managed Service pour Prometheus, peut vous être facturée. Vous pouvez également utiliser vos contrats de AWS support existants pour obtenir de l'aide.

En tant qu'administrateur de la plateforme Kubernetes et responsable financier, vous pouvez utiliser Kubecost pour visualiser le détail des frais Amazon EKS, répartir les coûts et facturer les unités organisationnelles telles que les équipes chargées des applications. Vous pouvez fournir à vos équipes internes et à vos unités commerciales des données de coûts transparentes et précises basées sur leur AWS facture réelle. En outre, vous pouvez également obtenir des recommandations personnalisées pour l'optimisation des coûts en fonction de leur environnement d'infrastructure et des modèles d'utilisation au sein de leurs clusters. Pour plus d'informations sur Kubecost, consultez la documentation de Kubecost.

Quelle est la différence entre le bundle personnalisé de Kubecost et la version gratuite de Kubecost (également connue sous le nom de) ? OpenCost

AWS et Kubecost ont collaboré pour proposer une version personnalisée de Kubecost. Cette version inclut un sous-ensemble de fonctions commerciales sans frais supplémentaires. Consultez les tableaux ci-dessous pour connaître les fonctionnalités incluses dans le bundle personnalisé de Kubecost.

Kubecost v2

Quelle est la différence entre Kubecost v1 et v2 ?

Kubecost 2.0 est une mise à niveau majeure par rapport aux versions précédentes et inclut de nouvelles fonctionnalités majeures, notamment un tout nouveau backend d'API. Notez que l'allocation et APIs les actifs sont entièrement rétrocompatibles. Consultez la documentation de Kubecost pour garantir une transition fluide. Pour la liste complète des améliorations, veuillez consulter l'annonce de Kubecost v2.0 et les notes de publication complètes.

> **Important**
>
> Consultez la documentation de Kubecost avant de procéder à la mise à niveau. La mise à niveau peut avoir un impact sur la disponibilité des rapports

Comparaison des fonctionnalités de base :

Fonctionnalité	Niveau 2.0 gratuit de Kubecost	Bundle Kubecost 2.0 optimisé pour Amazon EKS	Kubecost Entreprise 2.0
Visibilité des coûts du cluster	Clusters illimités jusqu'à 250 cœurs	Multicluster unifié sans limites de base lorsqu'il est intégré à Amazon Managed Service for Prometheus	Nombre unifié et illimité de clusters dans un nombre illimité d'environnements (par exemple, multi-cloud)
Déploiement	Hébergé par l'utilisateur	Hébergé par l'utilisateur	Hébergé par l'utilisateur, hébergé par Kubecost (locataire dédié), SaaS
Bases de données prises en charge	Prometheus local	Amazon Managed Service pour Prometheus ou Prometheus local	N'importe quelle version de Prometheus et bases de données personnalisées
Support pour la conservation des bases de données (mesures brutes)	15 jours	Données historiques illimitées	Données historiques illimitées
Rétention de l'API et de l'interface utilisateur (ETL) Kubecost	15 jours	15 jours	Illimité
Visibilité du cloud hybride	-	Clusters Amazon EKS et Amazon EKS Anywhere	Cloud multicloud et cloud hybride

Fonctionnalité	Niveau 2.0 gratuit de Kubecost	Bundle Kubecost 2.0 optimisé pour Amazon EKS	Kubecost Entreprise 2.0
Alertes et rapports récurrents	Uniquement pris en charge sur le cluster principal, limité à 250 cœurs	Alertes d'efficacité, alertes budgétaires, alertes de modificat ion des dépenses, etc., prises en charge dans tous les clusters	Alertes d'efficacité, alertes budgétaires, alertes de modificat ion des dépenses, etc., prises en charge dans tous les clusters
Rapports enregistrés	-	Rapports utilisant 15 jours de statistiques	Rapports utilisant un nombre illimité de données historiques et de métriques
Intégration de la facturation dans le cloud	Uniquement pris en charge sur le cluster principal, limité à 250 cœurs	Assistance tarifaire personnalisée pour AWS (y compris plusieurs clusters et plusieurs comptes)	Support tarifaire personnalisé pour tous les clouds
Recommandations en matière d'épargne	Uniquement pris en charge sur le cluster principal, limité à 250 cœurs	Informations sur le cluster principal, mais il n'y a pas de limite de 250 cœurs	Informations sur plusieurs clusters
Gouvernance : audits	-	-	Audit des événement s liés aux coûts historiques
Prise en charge de l'authentification unique (SSO)	-	Prise en charge d'Amazon Cognito	Okta, Auth0, PingID et tout autre élément KeyCloak personnal isé

Fonctionnalité	Niveau 2.0 gratuit de Kubecost	Bundle Kubecost 2.0 optimisé pour Amazon EKS	Kubecost Entreprise 2.0
Contrôle d'accès basé sur les rôles (RBAC) avec SAML 2.0	-	-	Okta, Auth0, PingID et tout autre élément KeyCloak personnalisé
Formation et intégration en entreprise	-	-	Service complet de formation et d'intégration FinOps
Equipes	-	-	Oui

Nouvelles fonctions :

Les fonctionnalités suivantes sont soumises à des limites métriques :

- Agrégateur Kubecost
- Surveillance du réseau
- Actions Kubecost
- Collections
- Détection des anomalies
- Dimensionnement correct de la demande de conteneur
- Prévisions Kubecost
- Autocomplétion pour le filtrage et l'agrégation

Limites métriques :

Métrique	Kubecost Free Tier 2.0	Bundle Kubecost 2.0 optimisé pour Amazon EKS	Kubecost Entreprise 2.0
Taille du cluster	Clusters illimités jusqu'à 250 cœurs	Illimité	Illimité
Rétention métrique	15 jours	15 jours	Illimité
Prise en charge de plusieurs clusters	Non disponible	Disponible	Disponible
Limites fondament ales	250 cœurs par cluster	Aucune limite de base	Aucune limite de base

Kubecost v1

Fonctionnalité	Niveau gratuit de Kubecost	Bundle Kubecost optimisé pour Amazon EKS	Kubecost Entreprise
Déploiement	Hébergé par l'utilisa teur	Hébergé par l'utilisa teur	Hébergé par l'utilisa teur ou hébergé par Kubecost (SaaS)
Nombre de clusters pris en charge	Illimité	Illimité	Illimité
Bases de données prises en charge	Prometheus local	Prometheus local ou Amazon Managed Service pour Prometheus	Prometheus, service géré par Amazon pour Prometheus, Cortex ou Thanos
Prise en charge de la conservation des bases de données	15 jours	Données historiques illimitées	Données historiques illimitées

Fonctionnalité	Niveau gratuit de Kubecost	Bundle Kubecost optimisé pour Amazon EKS	Kubecost Entreprise
Rétention des API Kubecost (ETL)	15 jours	15 jours	Données historiques illimitées
Visibilité des coûts du cluster	Clusters uniques	Multicluster unifié	Multicluster unifié
Visibilité du cloud hybride	-	Clusters Amazon EKS et Amazon EKS Anywhere	Support multicloud et cloud hybride
Alertes et rapports récurrents	-	Prise en charge des alertes d'efficac ité, des alertes de budget, des alertes de modification des dépenses, et bien d'autres encore	Prise en charge des alertes d'efficac ité, des alertes de budget, des alertes de modification des dépenses, et bien d'autres encore
Rapports enregistrés	-	Rapports utilisant des données sur 15 jours	Rapports utilisant des données historiques illimitées
Intégration de la facturation dans le cloud	Nécessaire pour chaque cluster individuel	Assistance tarifaire personnalisée pour AWS (y compris plusieurs clusters et plusieurs comptes)	Assistance tarifaire personnalisée pour AWS (y compris plusieurs clusters et plusieurs comptes)
Recommandations en matière d'épargne	Informations sur un seul cluster	Informations sur un seul cluster	Informations sur plusieurs clusters
Gouvernance : audits	-	-	Audit des événement s liés aux coûts historiques

Fonctionnalité	Niveau gratuit de Kubecost	Bundle Kubecost optimisé pour Amazon EKS	Kubecost Entreprise
Prise en charge de l'authentification unique (SSO)	-	Prise en charge d'Amazon Cognito	Okta, Auth0, PingID, KeyCloak
Contrôle d'accès basé sur les rôles (RBAC) avec SAML **2.0**	-	-	Okta, Auth0, PingID, Keycloak
Formation et intégration en entreprise	-	-	Service complet de formation et d'intégration FinOps

Questions fréquentes (FAQ)

Consultez les questions et réponses courantes suivantes sur l'utilisation de Kubecost avec Amazon EKS.

Qu'est-ce que la fonctionnalité de rétention des API (ETL) de Kubecost ?

La fonctionnalité ETL de Kubecost agrège et organise les métriques pour obtenir une visibilité des coûts à différents niveaux de granularité (tels que`namespace-level`, `pod-level` et). `deployment-level` Dans le cas du bundle Kubecost optimisé pour Amazon EKS, les clients obtiennent des données et des informations à partir des statistiques des 15 derniers jours.

Qu'est-ce que la fonctionnalité d'alertes et de rapports récurrents ? Quelles alertes et quels rapports inclut-il ?

Les alertes Kubecost permettent aux équipes de recevoir des mises à jour sur les dépenses en temps réel sur Kubernetes ainsi que sur les dépenses liées au cloud. Les rapports récurrents permettent aux équipes de recevoir des vues personnalisées de l'historique des dépenses liées à Kubernetes et au cloud. Les deux sont configurables à l'aide de l'interface utilisateur de Kubecost ou des valeurs Helm. Ils prennent en charge le courrier électronique, Slack et Microsoft Teams.

Que contiennent les rapports enregistrés ?

Les rapports enregistrés par Kubecost sont des vues prédéfinies des indicateurs de coûts et d'efficacité. Ils incluent le coût par cluster, l'espace de noms, l'étiquette et bien plus encore.

Qu'est-ce que l'intégration de la facturation dans le cloud ?

L'intégration à AWS la facturation APIs permet à Kubecost d'afficher out-of-cluster les coûts (comme Amazon S3). En outre, cela permet à Kubecost de concilier les prévisions de Kubecost relatives au cluster avec les données de facturation réelles afin de tenir compte de l'utilisation ponctuelle, des plans d'épargne et des remises accordées aux entreprises.

Que comprennent les recommandations en matière d'épargne ?

Kubecost fournit des informations et une automatisation pour aider les utilisateurs à optimiser leur infrastructure Kubernetes et leurs dépenses.

Cette fonctionnalité est-elle payante ?

Non Vous pouvez utiliser le bundle Kubecost optimisé pour Amazon EKS sans frais supplémentaires. Si vous souhaitez des fonctionnalités supplémentaires de Kubecost qui ne sont pas incluses, vous pouvez acheter une licence d'entreprise de Kubecost via le AWS Marketplace ou directement auprès de Kubecost.

Le support est-il disponible pour le bundle Kubecost optimisé pour Amazon EKS ?

Oui, uniquement si vous utilisez le bundle Kubecost optimisé pour Amazon EKS.

Comment puis-je obtenir de l'assistance pour le bundle Kubecost optimisé pour Amazon EKS ?

Vous pouvez ouvrir un dossier d'assistance auprès de l'équipe d' AWS assistance sur Contact AWS.

Ai-je besoin d'une licence pour utiliser les fonctionnalités de Kubecost fournies par l'intégration Amazon EKS ?

Non.

Puis-je intégrer Kubecost au rapport sur les AWS coûts et l'utilisation pour des rapports plus précis ?

Oui. Vous pouvez configurer Kubecost pour intégrer les données du rapport sur les AWS coûts et l'utilisation afin d'obtenir une visibilité précise des coûts, y compris les remises, les tarifs ponctuels, les prix des instances réservées, etc. Pour plus d'informations, consultez la section Intégration de la facturation dans le AWS cloud dans la documentation de Kubecost.

Cette version prend-elle en charge la gestion des coûts des clusters Kubernetes autogérés sur Amazon ? EC2

Non Le bundle Kubecost optimisé pour Amazon EKS est uniquement compatible avec les clusters Amazon EKS.

Kubecost peut-il suivre les coûts d'Amazon EKS sur Fargate ? AWS

Kubecost met tout en œuvre pour afficher la visibilité des coûts des clusters pour Amazon EKS sur Fargate, mais avec une précision moindre qu'avec Amazon EKS sur Amazon. EC2 Cela est principalement dû à la différence dans la façon dont vous êtes facturé pour votre utilisation. Avec Amazon EKS sur Fargate, les ressources consommées vous sont facturées. Avec Amazon EKS sur les EC2 nœuds Amazon, les ressources mises en service vous sont facturées. Kubecost calcule le coût d'un EC2 nœud Amazon en fonction de la spécification du nœud, qui inclut le processeur, la RAM et le stockage éphémère. Avec Fargate, les coûts sont calculés en fonction des ressources demandées pour les Fargate Pods.

Comment puis-je obtenir les mises à jour et les nouvelles versions de Kubecost ?

Vous pouvez mettre à niveau votre version de Kubecost à l'aide des procédures de mise à niveau standard de Helm. Les dernières versions se trouvent dans la Galerie publique Amazon ECR.

La **`kubectl-cost`** CLI est-elle prise en charge ? Comment l'installer ?

Oui `Kubectl-costest` un outil open source de Kubecost (licence Apache 2.0) qui fournit un accès CLI aux métriques de répartition des coûts de Kubernetes. Pour procéder à l'installation`kubectl-cost`, reportez-vous à la section Installation sur GitHub.

L'interface utilisateur de Kubecost est-elle prise en charge ? Comment y accéder ?

Kubecost fournit un tableau de bord Web auquel vous pouvez accéder via une redirection de `kubectl` port, une entrée ou un équilibreur de charge. Vous pouvez également utiliser le AWS Load Balancer Controller pour exposer Kubecost et utiliser Amazon Cognito pour l'authentification, les autorisations et la gestion des utilisateurs. Pour plus d'informations, consultez Comment utiliser Application Load Balancer et Amazon Cognito pour authentifier les utilisateurs de vos applications Web Kubernetes sur le blog. AWS

Fonctionnalités supplémentaires de Kubecost

- Les fonctionnalités suivantes sont disponibles dans Kubecost v1 et v2.

- Mesures des coûts d'exportation — La surveillance optimisée des coûts d'Amazon EKS est déployée avec Kubecost et Prometheus, qui sont un système de surveillance open source et une base de données de séries chronologiques. Kubecost lit les métriques de Prometheus, puis effectue des calculs de répartition des coûts et les réécrit dans Prometheus. Le front-end de Kubecost lit les métriques de Prometheus et les affiche sur l'interface utilisateur de Kubecost. L'architecture est illustrée dans le schéma suivant.

  Avec Prometheus préinstallé, vous pouvez écrire des requêtes pour intégrer les données Kubecost dans votre système de business intelligence actuel pour une analyse plus approfondie. Vous pouvez également l'utiliser comme source de données pour votre tableau de bord Grafana actuel afin d'afficher les coûts du cluster Amazon EKS que vos équipes internes connaissent bien. Pour en savoir plus sur la façon d'écrire des requêtes Prometheus, consultez le site https://opencost. io/docs/installation/prometheus/ GitHub ou `readme` utilisez les exemples de modèles JSON Grafana dans le référentiel Kubecost Github comme références.

- AWS Intégration des rapports sur les coûts et l'utilisation : pour effectuer des calculs de répartition des coûts pour votre cluster Amazon EKS, Kubecost extrait les informations de tarification publiques des AWS services et des AWS ressources à partir de l'API AWS Price List. Vous pouvez également intégrer Kubecost au rapport sur les AWS coûts et l'utilisation pour améliorer la précision des informations de tarification spécifiques à votre AWS compte. Ces informations incluent les programmes de remise pour les entreprises, l'utilisation d'instances réservées, les plans d'épargne et l'utilisation ponctuelle. Pour en savoir plus sur le fonctionnement de l'intégration des rapports sur les AWS coûts et l'utilisation, consultez la section Intégration de la facturation dans le AWS cloud dans la documentation de Kubecost.

Afficher l'utilisation des ressources avec le serveur Kubernetes Metrics

Le serveur Kubernetes Metrics est un agrégateur de données d'utilisation des ressources dans votre cluster, et il n'est pas déployé par défaut dans les clusters Amazon EKS. Pour plus d'informations, consultez Kubernetes Metrics Server sur. GitHub Le serveur Metrics est couramment utilisé par d'autres modules complémentaires de Kubernetes, tels que les déploiements de pods Scale avec Horizontal Pod Autoscaler ou le tableau de bord Kubernetes. Pour plus d'informations, consultez Pipeline de métriques de ressource dans la documentation Kubernetes. Cette rubrique explique comment déployer le serveur de métriques Kubernetes sur votre cluster Amazon EKS.

> **Important**
>
> Les métriques sont destinées à point-in-time l'analyse et ne constituent pas une source précise pour l'analyse historique. Ils ne peuvent pas être utilisés comme solution de surveillance ou à d'autres fins autres que le dimensionnement automatique. Pour de plus amples informations sur la surveillance, consultez Surveiller les clusters.

Considérations

- Si vous déployez manuellement le serveur Kubernetes Metrics sur des nœuds Fargate à l'aide du manifeste, configurez le `metrics-server` déploiement pour utiliser un port autre que le port par défaut de. `10250` Ce port est réservé à Fargate. La version complémentaire Amazon EKS de Metrics Server est préconfigurée pour utiliser le port`10251`.
- Assurez-vous que les groupes de sécurité et le réseau ACLs autorisent le port `10250` entre les `metrics-server` pods et tous les autres nœuds et pods. Le serveur de métriques Kubernetes utilise toujours le port `10250` pour collecter des métriques à partir d'autres points de terminaison du cluster. Si vous déployez sur des nœuds Fargate, autorisez à la fois le port et le port du serveur de mesures alternatifs configurés. `10250`

Déployez en tant que module complémentaire communautaire avec les modules complémentaires Amazon EKS

Nouveau : vous pouvez désormais déployer Metrics Server en tant que module complémentaire communautaire à l'aide de la AWS console ou d'Amazon EKS APIs.

Déploiement avec AWS console

1. Ouvrez votre cluster EKS dans la AWS console
2. Dans l'onglet « Modules complémentaires », sélectionnez Obtenir plus de modules complémentaires.
3. Dans la section « Extensions communautaires », sélectionnez Metrics Server, puis Next
4. EKS détermine la version appropriée du module complémentaire pour votre cluster. Vous pouvez modifier la version à l'aide du menu déroulant Version.
5. Sélectionnez Suivant puis Créer pour installer le module complémentaire.

Ressources supplémentaires

En savoir plus sur the section called "Modules complémentaires communautaires".

Vous installez ou mettez à jour les modules complémentaires communautaires de la même manière que les autres modules complémentaires Amazon EKS.

- the section called "Création d'un module complémentaire"
- the section called "Mettre à jour un module complémentaire"
- the section called "Supprimer un module complémentaire"

Déployer avec un manifeste

Nouveau : vous pouvez désormais déployer Metrics Server en tant que module complémentaire communautaire à l'aide de la AWS console ou d'Amazon EKS APIs. Ces instructions d'installation du manifeste seront archivées.

1. Déployez le serveur de métriques à l'aide de la commande suivante :

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/
   download/components.yaml
   ```

   Si vous utilisez Fargate, vous devez modifier ce fichier. Dans la configuration par défaut, le serveur de métriques utilise le port 10250. Ce port est réservé sur Fargate. Remplacez les références au port 10250 dans components.yaml par un autre port, tel que 10251.

2. Vérifiez que le `metrics-server` déploiement exécute le nombre de pods souhaité à l'aide de la commande suivante.

```
kubectl get deployment metrics-server -n kube-system
```

L'exemple qui suit illustre un résultat.

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

3. Testez le fonctionnement du serveur de métriques en affichant l'utilisation des ressources (CPU/mémoire) des nœuds.

```
kubectl top nodes
```

4. Si vous recevez le message d'erreur`Error from server (Forbidden)`, vous devez mettre à jour votre configuration Kubernetes RBAC. Votre identité Kubernetes RBAC nécessite des autorisations suffisantes pour lire les métriques du cluster. Vérifiez les autorisations d'API Kubernetes minimales requises pour lire les métriques sur. GitHub Découvrez comment accorder aux identités AWS IAM, telles que les rôles, l'accès à APIs Kubernetes.

Déployez des applications avec Helm sur Amazon EKS

Le gestionnaire de paquets Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes. Pour plus d'informations, consultez la documentation Helm. Cette rubrique vous aide à installer et à exécuter les fichiers binaires Helm, afin de pouvoir installer et gérer des charts à l'aide de la CLI Helm sur votre système local.

> **Important**
>
> Pour pouvoir installer des Charts Helm dans votre cluster, vous devez configurer `kubectl` pour qu'il fonctionne avec Amazon EKS. Si vous ne l'avez pas encore fait, veuillez consulter the section called "Cluster d'accès avec kubectl" avant de poursuivre. Si la commande suivante réussit pour votre cluster, vous êtes correctement configuré.
>
> ```
> kubectl get svc
> ```

1. Exécutez la commande adaptée à votre système d'exploitation client.
   - Si vous utilisez macOS avec Homebrew, installez les fichiers binaires à l'aide de la commande suivante.

     ```
     brew install helm
     ```

   - Pour plus d'options d'installation, voir Installation de Helm dans la documentation Helm.

     > **Note**
     >
     > Si vous recevez un message indiquant que `openssl` doit d'abord être installé, vous pouvez utiliser la commande suivante pour l'installer.

```
sudo yum install openssl
```

1. Pour récupérer le nouveau fichier binaire dans votre `PATH`, fermez votre fenêtre de terminal actuelle et ouvrez une nouvelle fenêtre.
2. Vérifiez la version de Helm que vous avez installée.

   ```
   helm version --template='{{ .Version }}{{ "\n" }}'
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v3.17.2
   ```

3. Assurez-vous que la version installée est compatible avec la version de votre cluster. Consultez Supported Version Skew pour en savoir plus. Par exemple, si vous utilisez une version de Kubernetes prise en charge`3.17.x`, elle ne doit pas être comprise entre ~. `1.29.x` `1.32.x`
4. À ce stade, vous pouvez exécuter toutes les commandes Helm (telles que `helm install` *`chart-name`*) pour installer, modifier, supprimer ou interroger les charts Helm dans votre cluster. Si vous utilisez Helm pour la première fois et que vous n'avez pas de charte spécifique à installer, vous pouvez :
   - Tester en installant un exemple de chart. Reportez-vous à la rubrique sur l'installation d'un exemple de chart dans le Guide Quickstart Helm.

- Créez un exemple de chart et envoyez-le à Amazon ECR. Pour plus d'informations, consultez la rubrique relative à l'envoi d'un chart Helm dans le Guide de l'utilisateur Amazon Elastic Container Registry.
- Installez un graphique Amazon EKS depuis le GitHub dépôt eks-charts ou depuis. ArtifactHub

Organisez les ressources Amazon EKS à l'aide de balises

Vous pouvez utiliser les balises pour vous aider à gérer vos ressources Amazon EKS. Cette rubrique fournit une vue d'ensemble de la fonction des identifications et indique comment créer des identifications.

Rubriques

- Principes de base des étiquettes
- Balisage de vos ressources
- Restrictions liées aux étiquettes
- Identification de vos ressources pour facturation
- Gestion des étiquettes à l'aide de la console
- Gestion des identifications à l'aide de la CLI, de l'API ou de eksctl

> **Note**
>
> Les balises sont un type de métadonnées distinct des étiquettes et des annotations Kubernetes. Pour plus d'informations sur ces autres types de métadonnées, consultez les sections suivantes dans la documentation Kubernetes :
>
> - Étiquettes et sélecteurs
> - Annotations

Principes de base des étiquettes

Une étiquette est une étiquette que vous attribuez à une AWS ressource. Chaque balise est constituée d'une clé et d'une valeur facultative.

Les tags vous permettent de classer vos AWS ressources par catégories. Par exemple, vous pouvez classer les ressources par objectif, propriétaire ou environnement. Lorsque vous avez de nombreuses ressources de même type, vous pouvez utiliser les balises que vous avez attribuées à une ressource spécifique pour identifier cette dernière rapidement. Par exemple, vous pouvez définir un ensemble de balises pour vos clusters Amazon EKS afin de vous aider à suivre le propriétaire et le niveau de pile de chaque cluster. Nous vous recommandons de concevoir un ensemble cohérent de clés de balise pour chaque type de ressource. Vous pouvez rechercher et filtrer les ressources en fonction des balises que vous ajoutez.

Une fois que vous avez ajouté une balise, vous pouvez modifier les clés et valeurs de balise ou supprimer les balises d'une ressource à tout moment. Si vous supprimez une ressource, ses balises sont également supprimées.

Les balises n'ont aucune signification sémantique pour Amazon EKS et sont interprétées strictement comme des chaînes de caractères. Vous pouvez définir la valeur d'une balise à une chaîne vide. Cependant, vous ne pouvez pas définir la valeur d'une balise sur null. Si vous ajoutez une balise ayant la même clé qu'une balise existante sur cette ressource, la nouvelle valeur remplace l'ancienne valeur.

Si vous utilisez AWS Identity and Access Management (IAM), vous pouvez contrôler quels utilisateurs de votre AWS compte sont autorisés à gérer les tags.

Balisage de vos ressources

Les ressources Amazon EKS suivantes prennent en charge les balises :

- clusters
- groupes de nœuds gérés
- profils Fargate

Vous pouvez baliser ces ressources à l'aide des éléments suivants :

- Si vous utilisez la console Amazon EKS, vous pouvez appliquer des balises à des ressources nouvelles ou existantes à tout moment. Pour ce faire, utilisez la commande Identifications sur la page de ressources correspondante. Pour de plus amples informations, veuillez consulter the section called "Gestion des étiquettes à l'aide de la console".
- Si vous utilisez`eksctl`, vous pouvez appliquer des balises aux ressources lors de leur création à l'aide de `--tags` cette option.

- Si vous utilisez la AWS CLI, l'API Amazon EKS ou un AWS SDK, vous pouvez appliquer des balises aux nouvelles ressources en utilisant le `tags` paramètre de l'action d'API correspondante. Vous pouvez également appliquer des identifications aux ressources à l'aide de l'action d'API `TagResource`. Pour de plus amples informations, veuillez consulter TagResource.

Lorsque vous exécutez certaines actions de création des ressources, vous pouvez également spécifier des balises pour la ressource en même temps que vous la créez. Si les balises ne peuvent pas être appliquées lors de la création de la ressource, celle-ci ne peut pas être créée. Ce mécanisme garantit que les ressources que vous prévoyez de baliser sont créées avec les balises que vous spécifiez ou ne sont pas créées du tout. Si vous balisez des ressources lorsque vous les créez, il n'est pas nécessaire d'exécuter des scripts de balisage personnalisés après avoir créé la ressource.

Les balises ne se propagent pas aux autres ressources associées à la ressource que vous créez. Par exemple, les balises de profil Fargate ne se propagent pas aux autres ressources associées au profil Fargate, telles que les pods planifiés avec celui-ci.

Restrictions liées aux étiquettes

Les restrictions suivantes s'appliquent aux balises :

- Un maximum de 50 balises peut être associé à une ressource.
- Les clés de balise ne peuvent pas être répétées pour une ressource. Chaque clé de balise doit être unique et ne peut avoir qu'une seule valeur.
- Les clés peuvent contenir jusqu'à 128 caractères en UTF-8.
- Les valeurs peuvent contenir jusqu'à 256 caractères en UTF-8.
- Si plusieurs AWS services et ressources utilisent votre schéma de balisage, limitez les types de caractères que vous utilisez. Certains services peuvent présenter des restrictions sur les caractères autorisés. Les caractères généralement autorisés sont les lettres, les chiffres, les espaces et les caractères suivants : `+ - = . _ : / @`.
- Les clés et valeurs de balise sont sensibles à la casse.
- N'utilisez pas `aws: AWS:`, ni aucune combinaison majuscules ou minuscules, comme un préfixe pour les clés ou les valeurs. Ils sont réservés uniquement à AWS l'usage. Vous ne pouvez pas modifier ou supprimer des clés ou des valeurs de balise avec ce préfixe. Les balises comportant ce préfixe ne sont pas prises en compte dans votre tags-per-resource limite.

Identification de vos ressources pour facturation

Lorsque vous appliquez des balises à des clusters Amazon EKS, vous pouvez les utiliser pour la répartition des coûts dans vos Cost & Usage Reports (Rapports sur les coûts et l'utilisation). Les données de mesure de vos rapports sur les coût et l'utilisation illustrent l'utilisation parmi tous vos clusters Amazon EKS. Pour plus d'informations, consultez le rapport sur les AWS coûts et l'utilisation dans le Guide AWS de l'utilisateur de facturation.

La balise de répartition des coûts AWS générée, en particulier`aws:eks:cluster-name`, vous permet de ventiler les coûts des EC2 instances Amazon par cluster Amazon EKS individuel dans Cost Explorer. Cependant, cette balise ne prend pas en compte les dépenses du plan de contrôle. La balise est automatiquement ajoutée aux EC2 instances Amazon qui participent à un cluster Amazon EKS. Ce comportement se produit indépendamment du fait que les instances soient provisionnées à l'aide de groupes de nœuds gérés par Amazon EKS, de Karpenter ou directement auprès d'Amazon. EC2 Cette identification n'est pas prise en compte dans la limite des 50 identifications. Pour utiliser le tag, le propriétaire du compte doit l'activer dans la console AWS de facturation ou à l'aide de l'API. Lorsqu'un titulaire du compte de gestion d'une AWS organisation active le tag, celui-ci est également activé pour tous les comptes des membres de l'organisation.

Vous pouvez également organiser vos informations de facturation en fonction des ressources possédant les mêmes valeurs de clé d'étiquette. Par exemple, vous pouvez étiqueter plusieurs ressources avec un nom d'application spécifique, puis organiser vos informations de facturation. De cette façon, vous pouvez afficher le coût total de cette application dans plusieurs services. Pour plus d'informations sur la configuration d'un rapport de répartition des coûts avec des balises, consultez le rapport de répartition des coûts mensuel dans le guide AWS de l'utilisateur de facturation.

> **Note**
>
> Si vous venez d'activer la création de rapports, les données du mois en cours peuvent être consultées après 24 heures.

Cost Explorer est un outil de reporting disponible dans le cadre du niveau AWS gratuit. Vous pouvez utiliser Cost Explorer pour consulter les graphiques de vos ressources Amazon EKS des 13 derniers mois. Vous pouvez également prévoir le montant que vous êtes susceptible de dépenser au cours des trois prochains mois. Vous pouvez observer des tendances dans le montant que vous dépensez en AWS ressources au fil du temps. Par exemple, vous pouvez l'utiliser pour identifier les zones qui méritent d'être approfondies et connaître les tendances que vous pouvez utiliser pour comprendre

vos coûts. Vous pouvez également spécifier des plages de temps pour les données et afficher des données temporelles par jour ou par mois.

Gestion des étiquettes à l'aide de la console

À l'aide de la console Amazon EKS, vous pouvez gérer les identifications qui sont associées à des clusters et groupes de nœuds gérés nouveaux ou existants.

Lorsque vous sélectionnez une page spécifique aux ressources dans la console Amazon EKS, cette page affiche une liste de ces ressources. Par exemple, si vous sélectionnez Clusters dans le panneau de navigation de gauche, la console affiche une liste de clusters Amazon EKS. Lorsque vous sélectionnez une ressource de l'une de ces listes (par exemple, un cluster spécifique) qui prend en charge les balises, vous pouvez afficher et gérer ses balises dans l'onglet Tags (Balises).

Vous pouvez également utiliser l'éditeur de balises dans le AWS Management Console, qui fournit un moyen unifié de gérer vos balises. Pour plus d'informations, consultez la section Marquage de vos AWS ressources avec l'éditeur de balises dans le guide de l'utilisateur de l'éditeur de AWS balises.

Ajout de balises lors de la création d'une ressource

Vous pouvez ajouter des identifications aux clusters, groupes de nœuds gérés et profils Fargate Amazon EKS lorsque vous les créez. Pour de plus amples informations, veuillez consulter the section called “Créer un cluster ”.

Ajout et suppression de balises sur une ressource

Vous pouvez ajouter ou supprimer les balises associées à vos clusters directement depuis la page de la ressource.

1. Ouvrez la console Amazon EKS.
2. Dans la barre de navigation, sélectionnez la AWS région à utiliser.
3. Dans le volet de navigation de gauche, choisissez Clusters.
4. Choisissez un cluster spécifique.
5. Choisissez l'onglet Tags (Identifications), puis Manage tags (Gérer les identifications).
6. Sur la page Gérer les balises, ajoutez ou supprimez vos balises si nécessaire.
   - Pour ajouter une balise, choisissez Ajouter une balise. Puis spécifiez la clé et la valeur de chaque balise.

- Pour supprimer une balise, sélectionnez Remove tag (Supprimer une balise).

7. Répétez cette procédure pour chaque balise que vous voulez ajouter ou supprimer.
8. Choisissez Update (Mettre à jour) pour terminer.

Gestion des identifications à l'aide de la CLI, de l'API ou de `eksctl`

Utilisez les commandes AWS CLI ou les opérations d'API Amazon EKS suivantes pour ajouter, mettre à jour, répertorier et supprimer les balises de vos ressources. Vous pouvez uniquement utiliser `eksctl` pour ajouter des balises tout en créant simultanément les nouvelles ressources avec une seule commande.

Tâche	AWS CLI	AWS Outils pour Windows PowerShell	Action d'API
Ajouter ou remplacer une ou plusieurs étiquettes.	tag-resource	Ajouter une EKSResource étiquette	TagResource
Supprimer une ou plusieurs étiquettes.	untag-resource	Supprimer-EKSResource Tag	UntagResource

Les exemples suivants montrent comment baliser ou débaliser des ressources à l'aide de la AWS CLI.

Exemple 1 : Étiqueter un cluster existant

La commande suivante permet d'étiqueter un cluster existant.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Exemple 2 : Supprimer les étiquettes d'un cluster existant

La commande suivante permet de supprimer une étiquette d'un cluster existant.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Exemple 3 : Afficher la liste des étiquettes d'une ressource

La commande suivante permet de répertorier l'ensemble des balises qui sont associées à une ressource existante.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Lorsque vous utilisez certaines actions de création de ressources, vous pouvez spécifier des balises en même temps que vous créez la ressource. Les actions suivantes prennent en charge la spécification d'une balise lorsque vous créez une ressource.

Tâche	AWS CLI	AWS Outils pour Windows PowerShell	Action d'API	eksctl
Créer un cluster	create-cluster	Nouveau-EKSCluster	CreateCluster	`create cluster`
Créer un groupe de nœuds géré*	create-nodegroup	Nouveau-EKSNodegroup	CreateNodegroup	`create nodegroup`
Créer un profil Fargate	create-fargate-profile	Nouveau-EKSFargateProfil	CreateFargateProfile.html	`create fargateprofile`

- Si vous souhaitez également étiqueter les EC2 instances Amazon lorsque vous créez un groupe de nœuds gérés, créez le groupe de nœuds gérés à l'aide d'un modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Marquage des instances Amazon EC2 ". Si vos instances existent déjà, vous pouvez étiqueter manuellement les instances. Pour plus d'informations, consultez la section Marquage de vos ressources dans le guide de l' EC2 utilisateur Amazon.

Afficher et gérer les quotas de service Amazon EKS et Fargate

Amazon EKS a intégré Service Quotas, un AWS service que vous pouvez utiliser pour consulter et gérer vos quotas depuis un emplacement central. Pour plus d'informations, veuillez consulter Qu'est-ce que Service Quotas? dans le Guide de l'utilisateur Service Quotas. Grâce à l'intégration des

quotas de service, vous pouvez rapidement rechercher la valeur de vos quotas de service Amazon EKS et AWS Fargate à l'aide de la AWS Management Console CLI and. AWS

Consultez les quotas de service EKS dans le AWS Management Console

1. Ouvrez la console Service Quotas.
2. Dans le volet de navigation de gauche, sélectionnez AWS services.
3. Dans la liste des AWS services, recherchez et sélectionnez Amazon Elastic Kubernetes Service (Amazon EKS) ou Fargate. AWS

   Dans la liste des quotas de service, vous pouvez voir le nom du quota de service, la valeur appliquée (si elle est disponible), le quota AWS par défaut et si la valeur du quota est ajustable.
4. Pour afficher des informations supplémentaires sur un quota de service, notamment la description, choisissez le nom du quota.
5. (Facultatif) Pour demander une augmentation de quota, sélectionnez le quota que vous souhaitez augmenter, sélectionnez Request quota increase (Demander une augmentation de quota), saisissez ou sélectionnez les informations requises, puis sélectionnez Request (Demander).

Pour mieux utiliser les quotas de service à l'aide du AWS Management Console, consultez le Guide de l'utilisateur des quotas de service. Pour demander une augmentation de quota, consultez Demander une augmentation de quota dans le Guide de l'utilisateur de Service Quotas.

Afficher les quotas de service EKS avec la AWS CLI

Exécutez la commande suivante pour afficher vos quotas Amazon EKS.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

Exécutez la commande suivante pour afficher vos quotas Fargate.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
```

```
    --output table
```

> **Note**
>
> Le quota renvoyé est le nombre de tâches Amazon ECS ou de pods Amazon EKS qui peuvent être exécutés simultanément sur Fargate dans ce compte dans la région actuelle. AWS

Pour travailler davantage avec les quotas de service à l'aide de la AWS CLI, consultez service-quotas dans le manuel de référence des commandes de la AWS CLI. Pour demander une augmentation du quota, consultez la request-service-quota-increasecommande dans le manuel de référence des commandes de la AWS CLI.

Service quotas Amazon EKS

AWS recommande d'utiliser le AWS Management Console pour consulter vos quotas actuels. Pour de plus amples informations, veuillez consulter the section called “Consultez les quotas de service EKS dans le AWS Management Console”.

Pour consulter les quotas de service EKS par défaut, consultez la section Points de terminaison et quotas Amazon Elastic Kubernetes Service dans le manuel de référence général. AWS

Ces quotas de service sont répertoriés sous Amazon Elastic Kubernetes Service (Amazon EKS) dans la console Service Quotas. Pour demander une augmentation du quota pour les valeurs affichées comme ajustables, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.

> **Note**
>
> Les ajustements apportés aux composants suivants ne sont pas pris en charge dans les Service Quotas :* Associations d'identité de pod par cluster. Pour les limites, voirthe section called “Identité du pod”. * CIDRs pour les réseaux de nœuds distants ou les réseaux de ports distants pour les nœuds hybrides. Pour les limites, voirthe section called “Nœuds hybrides”.

AWS Quotas de service Fargate

Le service AWS Fargate de la console Service Quotas répertorie plusieurs quotas de service. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche d'un quota de service. Pour de plus amples informations, veuillez consulter the section called "Création d'une CloudWatch alarme pour surveiller les métriques d'utilisation des ressources de Fargate".

AWS Les nouveaux comptes peuvent avoir des quotas initiaux plus faibles qui peuvent augmenter au fil du temps. Fargate surveille en permanence l'utilisation du compte dans AWS chaque région, puis augmente automatiquement les quotas en fonction de l'utilisation. Vous pouvez également demander une augmentation du quota pour les valeurs désignées comme ajustables. Pour de plus amples informations, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.

AWS recommande d'utiliser le AWS Management Console pour consulter vos quotas actuels. Pour de plus amples informations, veuillez consulter the section called "Consultez les quotas de service EKS dans le AWS Management Console".

Pour consulter les quotas de service AWS Fargate par défaut sur EKS, consultez la section Quotas de service Fargate dans le manuel de référence général. AWS

> **Note**
>
> Fargate applique également les tâches Amazon ECS et les quotas de taux de lancement des Amazon EKS Pods. Pour plus d'informations, consultez la section « AWS Fargate throttling quotas » dans le guide Amazon ECS.

Sécurité dans Amazon EKS

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. Pour Amazon EKS, AWS est responsable du plan de contrôle Kubernetes, qui inclut les nœuds et la base de données du plan de contrôle. `etcd` Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon EKS, consultez Services AWS concernés par le programme de conformité.
- Sécurité dans le cloud : votre responsabilité englobe les domaines suivants :
  - La configuration de sécurité du plan de données, y compris la configuration des groupes de sécurité qui autorisent le trafic à transmettre à partir du plan de contrôle Amazon EKS dans le VPC client.
  - La configuration des nœuds et les conteneurs eux-mêmes
  - Le système d'exploitation du nœud (y compris les mises à jour et les correctifs de sécurité)
  - D'autres logiciels d'application connexes :
    - Configuration et gestion des contrôles réseau, tels que les règles de pare-feu
    - La gestion de l'identité au niveau de la plateforme et la gestion des accès, avec ou en complément de l'IAM
  - La sensibilité de vos données, les exigences de votre entreprise,et la législation et la réglementation applicables

Amazon EKS est certifié par plusieurs programmes de conformité pour les applications réglementées et sensibles. Amazon EKS est conforme aux normes SOC, PCI, ISO, FedRAMP-Moderate, IRAP, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF et est un service éligible à la loi HIPAA. Pour de plus amples informations, veuillez consulter Gérez l'accès.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon EKS. Les rubriques suivantes vous montrent comment

configurer Amazon EKS pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources Amazon EKS.

> **Note**
>
> Les conteneurs Linux sont constitués de groupes de contrôle (cgroups) et d'espaces de noms qui aident à limiter les accès auxquels un conteneur peut accéder, mais tous les conteneurs partagent le même noyau Linux que l'instance Amazon EC2 hôte. L'exécution d'un conteneur en tant qu'utilisateur racine (UID 0) ou l'octroi d'un accès à un conteneur aux ressources hôtes ou aux espaces de noms tels que le réseau hôte ou l'espace de noms PID hôte sont fortement déconseillés, car cela réduit l'efficacité de l'isolation fournie par les conteneurs.

Rubriques

- Sécurisez les clusters Amazon EKS à l'aide des meilleures pratiques
- Analyser les vulnérabilités dans Amazon EKS
- Validation de conformité pour les clusters Amazon EKS
- Considérations relatives à la sécurité pour Amazon Elastic Kubernetes Service
- Considérations relatives à la sécurité pour Kubernetes
- Considérations relatives à la sécurité pour le mode automatique d'Amazon EKS
- Gestion des identités et des accès pour Amazon EKS

Sécurisez les clusters Amazon EKS à l'aide des meilleures pratiques

Les meilleures pratiques de sécurité d'Amazon EKS figurent dans les meilleures pratiques en matière de sécurité du guide des meilleures pratiques d'Amazon EKS.

Analyser les vulnérabilités dans Amazon EKS

La sécurité est une considération cruciale pour la configuration et la maintenance des clusters et des applications Kubernetes. Vous trouverez ci-dessous la liste des ressources vous permettant d'analyser la configuration de sécurité de vos clusters EKS, des ressources vous permettant de

détecter les vulnérabilités et des intégrations avec des AWS services capables de réaliser cette analyse à votre place.

Le test de référence du Center for Internet Security (CIS) pour Amazon EKS

Le test Kubernetes du Center for Internet Security (CIS) fournit des conseils pour les configurations de sécurité Amazon EKS. Le benchmark :

- S'applique aux EC2 nœuds Amazon (gérés et autogérés) où vous êtes responsable des configurations de sécurité des composants Kubernetes.
- Fournit un moyen standard approuvé par la communauté de vous assurer que vous avez configuré votre cluster et vos nœuds Kubernetes en toute sécurité lors de l'utilisation d'Amazon EKS.
- Se compose de quatre sections : configuration de journalisation du plan de contrôle, configurations de sécurité des nœuds, politiques et services managés.
- Prend en charge toutes les versions de Kubernetes actuellement disponibles dans Amazon EKS et peut être exécuté à l'aide de kube-bench, un outil open source standard pour vérifier la configuration à l'aide du benchmark CIS sur les clusters Kubernetes.

Pour en savoir plus, consultez Présentation de la norme CIS Amazon EKS.

Pour un `aws-sample` pipeline automatisé permettant de mettre à jour votre groupe de nœuds avec une AMI référencée CIS, voir Pipeline de renforcement des AMI optimisé pour EKS.

Versions de la plateforme Amazon EKS

Les versions de la plateforme Amazon EKS représentent les fonctionnalités du plan de contrôle du cluster, notamment les indicateurs du serveur d'API Kubernetes activés et la version actuelle du correctif Kubernetes. Les nouveaux clusters sont déployés avec la dernière version de la plateforme. Pour plus de détails, consultez le eks/latest/userguide/platform fichier -versions.html [EKS platform-versions, type="documentation "].

Vous pouvez mettre à jour un cluster Amazon EKS vers de nouvelles versions Kubernetes. À mesure que de nouvelles versions Kubernetes deviennent disponibles dans Amazon EKS, nous vous recommandons de mettre à jour de manière proactive vos clusters pour utiliser la dernière version disponible. Pour plus d'informations sur les versions de Kubernetes dans EKS, consultez le fichier -versions.html eks/latest/userguide/kubernetes [Versions prises en charge par Amazon EKS, type="documentation "].

Liste des vulnérabilités du système d'exploitation

AL2Liste des vulnérabilités 023

Suivez les événements liés à la sécurité ou à la confidentialité d'Amazon Linux 2023 dans le centre de sécurité Amazon Linux ou abonnez-vous au flux RSS associé. Les événements de sécurité et de confidentialité incluent une présentation du problème, des packages et des instructions relatives à la mise à jour de vos instances pour résoudre le problème.

Liste des vulnérabilités d'Amazon Linux 2

Suivez les évènements de sécurité et de confidentialité pour Amazon Linux 2 via le centre de sécurité Amazon Linux ou souscrivez au flux RSS associé. Les événements de sécurité et de confidentialité incluent une présentation du problème, des packages et des instructions relatives à la mise à jour de vos instances pour résoudre le problème.

Détection de nœuds avec Amazon Inspector

Vous pouvez utiliser Amazon Inspector pour vérifier l'accessibilité involontaire de vos nœuds au réseau et les vulnérabilités de ces EC2 instances Amazon.

Détection de clusters et de nœuds avec Amazon GuardDuty

Service de détection des GuardDuty menaces Amazon qui aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre AWS environnement. Parmi les autres fonctionnalités, GuardDuty propose les deux fonctionnalités suivantes qui détectent les menaces potentielles pour vos clusters EKS : la protection EKS et la surveillance du temps d'exécution.

Pour de plus amples informations, veuillez consulter the section called “Amazon GuardDuty”.

Validation de conformité pour les clusters Amazon EKS

Pour savoir si un AWS service entre dans le champ d'application de programmes de conformité spécifiques, consultez les AWS services dans Étendue par programme de conformité et choisissez le programme de conformité qui vous intéresse. Pour obtenir des informations générales, consultez Programmes de conformitéAWS.

Vous pouvez télécharger des rapports d'audit tiers à l'aide d' AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact.

Lorsque vous utilisez AWS des services, votre responsabilité en matière de conformité dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Conformité et gouvernance de la sécurité : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- Référence des services éligibles HIPAA : liste les services éligibles HIPAA. Tous les AWS services ne sont pas éligibles à la loi HIPAA.
- AWS Ressources relatives à la conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière de sécurisation AWS des services et présentent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- Évaluation des ressources à l'aide des règles du AWS guide du développeur de configuration : le service AWS Config évalue dans quelle mesure vos configurations de ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub — Ce AWS service fournit une vue complète de l'état de votre sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos AWS ressources et vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Ce AWS service détecte les menaces potentielles qui pèsent sur vos AWS comptes, vos charges de travail, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte ou malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- AWS Audit Manager : ce AWS service vous aide à auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Considérations relatives à la sécurité pour Amazon Elastic Kubernetes Service

Les considérations suivantes concernent la sécurité du cloud, dans la mesure où elles affectent Amazon EKS.

Rubriques

- Sécurité de l'infrastructure dans Amazon EKS
- Comprendre la résilience dans les clusters Amazon EKS

Sécurité de l'infrastructure dans Amazon EKS

En tant que service géré, Amazon Elastic Kubernetes Service est protégé par la sécurité du réseau mondial. AWS Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon EKS via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser le AWS Security Token Service (AWS STS) pour générer des informations de sécurité temporaires afin de signer les demandes.

Lorsque vous créez un cluster Amazon EKS, vous spécifiez les sous-réseaux VPC à utiliser par votre cluster. Amazon EKS nécessite des sous-réseaux dans au moins deux zones de disponibilité. Nous recommandons un VPC avec des sous-réseaux publics et privés afin que Kubernetes puisse créer des équilibreurs de charge publics dans les sous-réseaux publics qui équilibrent la charge du trafic vers les pods exécutés sur des nœuds situés dans des sous-réseaux privés.

Pour plus d'informations sur les considérations VPC, consultez the section called "Exigences requises pour le VPC et les sous-réseaux".

Si vous créez votre VPC et vos groupes de nœuds à l'aide des AWS CloudFormation modèles fournis dans la procédure pas à pas de prise en main d'Amazon EKS, votre plan de contrôle et vos groupes de sécurité de nœuds sont configurés selon nos paramètres recommandés.

Pour plus d'informations sur les considérations des groupes de sécurité, consultez the section called "Exigences de groupe de sécurité".

Lorsque vous créez un cluster, Amazon EKS crée un point de terminaison pour le serveur d'API Kubernetes géré que vous utilisez pour communiquer avec votre cluster (à l'aide d'outils de gestion Kubernetes comme `kubectl`). Par défaut, ce point de terminaison du serveur d'API est public sur Internet, et l'accès au serveur d'API est sécurisé à l'aide d'une combinaison d' AWS Identity and Access Management (IAM) et de contrôle d'accès basé sur le rôle (RBAC) Kubernetes natif.

Vous pouvez activer l'accès privé au serveur d'API Kubernetes pour que toutes les communications entre vos nœuds et le serveur d'API restent au sein de votre VPC. Vous pouvez limiter les adresses IP qui peuvent accéder à votre serveur API à partir d'Internet, ou désactiver complètement l'accès Internet au serveur d'API.

Pour plus d'informations sur la modification de l'accès au point de terminaison de cluster, consultez the section called "Modification de l'accès au point de terminaison de cluster".

Vous pouvez implémenter des politiques réseau Kubernetes avec Amazon VPC CNI ou des outils tiers tels que Project Calico. Pour plus d'informations sur l'utilisation de l'Amazon VPC CNI pour les stratégies réseau, consultez the section called "Politiques Kubernetes". Project Calico tiers est un projet open source tierce. Pour plus d'informations, consultez la documentation Project Calico.

Accédez à Amazon EKS à l'aide de AWS PrivateLink

Vous pouvez l'utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et Amazon Elastic Kubernetes Service. Vous pouvez accéder à Amazon EKS comme s'il se trouvait dans votre VPC, sans passer par une passerelle Internet, un appareil NAT, une connexion VPN ou une connexion Direct AWS Connect. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour accéder à Amazon EKS.

Vous établissez cette connexion privée en créant un point de terminaison d'interface optimisé par AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-

réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par le demandeur qui servent de point d'entrée pour le trafic destiné à Amazon EKS.

Pour plus d'informations, consultez la section Accès aux AWS services AWS PrivateLink dans le AWS PrivateLink Guide.

Avant de commencer

Avant de commencer, assurez-vous d'avoir effectué les tâches suivantes :

- Consultez la section Accès à un AWS service à l'aide d'un point de terminaison VPC d'interface dans le Guide AWS PrivateLink

Considérations

- Support et limites : les points de terminaison de l'interface Amazon EKS permettent un accès sécurisé à toutes les actions de l'API Amazon EKS depuis votre VPC, mais comportent des limitations spécifiques : ils ne prennent pas en charge l'accès à APIs Kubernetes, car ils disposent d'un point de terminaison privé distinct, vous ne pouvez pas configurer Amazon EKS pour qu'il soit accessible uniquement via le point de terminaison de l'interface.
- Tarification : L'utilisation de points de terminaison d'interface pour Amazon EKS entraîne des AWS PrivateLink frais standard : frais horaires pour chaque point de terminaison approvisionné dans chaque zone de disponibilité, frais de traitement des données pour le trafic via le point de terminaison. Pour en savoir plus, consultez la page Tarification AWS PrivateLink .
- Sécurité et contrôle d'accès : nous recommandons d'améliorer la sécurité et de contrôler l'accès grâce à ces configurations supplémentaires : utilisez des politiques de point de terminaison VPC pour contrôler l'accès à Amazon EKS via le point de terminaison de l'interface, associez des groupes de sécurité aux interfaces réseau des points de terminaison pour gérer le trafic, utilisez les journaux de flux VPC pour capturer et surveiller le trafic IP à destination et en provenance des points de terminaison de l'interface, les journaux pouvant être publiés sur Amazon ou Amazon S3. CloudWatch Pour en savoir plus, consultez Contrôler l'accès aux points de terminaison VPC à l'aide de politiques relatives aux points de terminaison et Journalisation du trafic IP à l'aide des journaux de flux VPC.
- Options de connectivité : les points de terminaison d'interface offrent des options de connectivité flexibles utilisant un accès sur site (connectez votre centre de données sur site à un VPC avec le point de terminaison d'interface via Direct AWS Connect ou AWS Site-to-Site VPN) ou via une connectivité inter-VPC (utilisez AWS Transit Gateway ou le peering VPC pour connecter une VPCs

autre personne au VPC avec le point de terminaison d'interface, en maintenant le trafic sur le réseau). AWS

- Support des versions IP : les points de terminaison créés avant août 2024 ne sont compatibles qu' IPv4 avec eks.region.amazonaws.com. Les nouveaux points de terminaison créés après août 2024 sont compatibles avec la technologie Dual-Stack IPv4 IPv6 (par exemple, eks.region.amazonaws.com, eks.region.api.aws).
- Disponibilité régionale : car AWS PrivateLink l'API EKS n'est pas disponible dans les régions Asie-Pacifique (Malaisie) (ap-southeast-5), Asie-Pacifique (Thaïlande) (ap-southeast-7), Mexique (centre) (mx-central-1) et Asie-Pacifique (Taipei) (ap-east-2). AWS PrivateLink le support pour eks-auth (EKS Pod Identity) est disponible dans la région Asie-Pacifique (Malaisie) (ap-southeast-5).

Créer un point de terminaison d'interface pour Amazon EKS

Vous pouvez créer un point de terminaison d'interface pour Amazon EKS à l'aide de la console Amazon VPC ou de l'interface de ligne de AWS commande (AWS CLI). Pour plus d'informations, consultez la section Créer un point de terminaison VPC dans le AWS PrivateLink Guide.

Créez un point de terminaison d'interface pour Amazon EKS à l'aide des noms de service suivants :

API EKS

- com.amazonaws.region-code.eks
- com.amazonaws.region-code.eks-fips (pour les terminaux conformes à la norme FIPS)

API d'authentification EKS (identité du pod EKS)

- com.amazonaws.region-code.eks-auth

Fonctionnalité DNS privée pour les points de terminaison de l'interface Amazon EKS

La fonctionnalité DNS privé, activée par défaut pour les points de terminaison d'interface d'Amazon EKS et d'autres AWS services, facilite les demandes d'API sécurisées et privées à l'aide des noms DNS régionaux par défaut. Cette fonctionnalité garantit que les appels d'API sont acheminés via le point de terminaison de l'interface via le AWS réseau privé, améliorant ainsi la sécurité et les performances.

La fonctionnalité DNS privé s'active automatiquement lorsque vous créez un point de terminaison d'interface pour Amazon EKS ou d'autres AWS services. Pour l'activer, vous devez configurer correctement votre VPC en définissant des attributs spécifiques :

- enableDnsHostnames: autorise les instances du VPC à avoir des noms d'hôte DNS.
- enableDnsSupport: active la résolution DNS dans l'ensemble du VPC.

Pour step-by-step obtenir des instructions sur la vérification ou la modification de ces paramètres, consultez Afficher et mettre à jour les attributs DNS de votre VPC.

Noms DNS et types d'adresses IP

Lorsque la fonctionnalité DNS privé est activée, vous pouvez utiliser des noms DNS spécifiques pour vous connecter à Amazon EKS, et ces options évoluent au fil du temps :

- eks.region.amazonaws.com : nom DNS traditionnel, résolu uniquement pour les adresses antérieures à août 2024. IPv4 Pour les points de terminaison existants mis à jour vers Dual-Stack, ce nom correspond à la fois aux adresses IPv4 et IPv6 aux adresses.
- eks.region.api.aws : Disponible pour les nouveaux points de terminaison créés après août 2024, ce nom DNS à double pile correspond à la fois aux adresses et aux adresses. IPv4 IPv6

Après août 2024, les nouveaux points de terminaison d'interface sont dotés de deux noms DNS, et vous pouvez opter pour le type d'adresse IP à double pile. Pour les points de terminaison existants, la mise à jour vers Dual-Stack modifie eks.region.amazonaws.com pour prendre en charge les deux et. IPv4 IPv6

Utilisation de la fonctionnalité DNS privé

Une fois configurée, la fonctionnalité DNS privé peut être intégrée à vos flux de travail, offrant les fonctionnalités suivantes :

- Demandes d'API : utilisez les noms DNS régionaux par défaut `eks.region.amazonaws.com` ou`eks.region.api.aws`, en fonction de la configuration de votre terminal, pour envoyer des demandes d'API à Amazon EKS.
- Compatibilité des applications : vos applications existantes qui appellent EKS ne APIs nécessitent aucune modification pour tirer parti de cette fonctionnalité.

- AWS CLI avec double pile : pour utiliser les points de terminaison à double pile avec la AWS CLI, consultez la configuration des points de terminaison à double pile et FIPS dans le guide de référence des outils et. AWS SDKs
- Routage automatique : tout appel vers le point de terminaison du service par défaut Amazon EKS est automatiquement dirigé via le point de terminaison de l'interface, garantissant ainsi une connectivité privée et sécurisée.

Comprendre la résilience dans les clusters Amazon EKS

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Amazon EKS exécute et fait évoluer le plan de contrôle Kubernetes sur plusieurs zones de AWS disponibilité afin de garantir une haute disponibilité. Amazon EKS met automatiquement à l'échelle les instances du plan de contrôle en fonction de la charge, détecte et remplace les instances du plan de contrôle non saines, et applique automatiquement des correctifs au plan de contrôle. Après que vous ayez lancé une mise à jour de version, Amazon EKS met à jour votre plan de contrôle pour vous, en maintenant la haute disponibilité du plan de contrôle pendant la mise à jour.

Ce plan de contrôle comprend au moins deux instances de serveur d'API et trois `etcd` instances réparties sur trois zones de disponibilité au sein d'une AWS région. Amazon EKS :

- Surveille activement la charge sur les instances de plan de contrôle et les met automatiquement à l'échelle pour garantir des performances élevées.
- Détecte et remplace automatiquement les instances du plan de contrôle défaillantes, en les redémarrant dans les zones de disponibilité de la AWS région selon les besoins.
- Exploite l'architecture des AWS régions afin de maintenir une haute disponibilité. De ce fait, Amazon EKS peut offrir un accord de niveau de service (SLA) pour la disponibilité des points de terminaison de serveur d'API.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS globale.

Considérations relatives à la sécurité pour Kubernetes

Les considérations suivantes concernent la sécurité dans le cloud, car elles affectent Kubernetes dans les clusters Amazon EKS. Pour un examen approfondi des contrôles et pratiques de sécurité dans Kubernetes, consultez Cloud Native Security et Kubernetes dans la documentation de Kubernetes.

Rubriques

- Sécurisez les charges de travail grâce aux certificats Kubernetes
- Comprendre les rôles et les utilisateurs RBAC créés par Amazon EKS
- Chiffrez les secrets Kubernetes avec KMS sur les clusters existants
- Utiliser AWS les secrets de Secrets Manager avec les pods Amazon EKS
- Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes

Sécurisez les charges de travail grâce aux certificats Kubernetes

L'API Kubernetes Certificates automatise le provisionnement des informations d'identification X.509. L'API comporte une interface de ligne de commande permettant aux clients de l'API Kubernetes de demander et d'obtenir des certificats X.509 auprès d'une autorité de certification (CA). Vous pouvez utiliser la ressource `CertificateSigningRequest` (CSR) pour demander à un signataire désigné de signer le certificat. Vos demandes sont approuvées ou refusées avant d'être signées. Kubernetes prend en charge à la fois les signataires intégrés et les signataires personnalisés avec des comportements bien définis. De cette façon, les clients peuvent prévoir ce qu'il adviendra de leur CSRs. Pour en savoir plus sur la signature des certificats, consultez les demandes de signature.

L'un des signataires intégrés est `kubernetes.io/legacy-unknown`. L'API `v1beta1` de la ressource CSR a honoré ce signataire d'héritage inconnu. Cependant, l'`v1`API stable de CSR ne permet pas `signerName` de définir le sur`kubernetes.io/legacy-unknown`.

Si vous souhaitez utiliser Amazon EKS CA pour générer des certificats sur vos clusters, vous devez utiliser un signataire personnalisé. Pour utiliser la version de l'API CSR `v1` et générer un nouveau certificat, vous devez migrer tous les manifestes et clients API existants. Les certificats existants créés avec l'API `v1beta1` existante sont valides et fonctionnent jusqu'à l'expiration du certificat. Cela inclut les éléments suivants :

- Distribution d'approbation : aucune. Il n'existe aucune confiance ou distribution standard pour ce signataire dans un cluster Kubernetes.

- Sujets autorisés : tous
- Extensions x509 autorisées : honore subjectAltName et utilise les extensions clés et supprime les autres extensions
- Utilisations de clés autorisées : ne doit pas inclure les utilisations au-delà de [« chiffrement de clé », « signature numérique », « authentification du serveur »]

> **Note**
>
> La signature de certificat client n'est pas prise en charge.

- Expiration/durée de vie du certificat : 1 an (par défaut et maximum)
- Bit CA autorisé/interdit : non autorisé

Exemple de génération CSR avec SignerName

Ces étapes montrent comment générer un certificat de service pour un nom DNS `myserver.default.svc` en utilisant `signerName: beta.eks.amazonaws.com/app-serving`. Utilisez-le comme guide pour votre propre environnement.

1. Exécutez la commande `openssl genrsa -out myserver.key 2048` pour générer une clé privée RSA.

   ```
   openssl genrsa -out myserver.key 2048
   ```

2. Utilisez la commande suivante pour générer une demande de certificat.

   ```
   openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
   ```

3. Générez une valeur `base64` pour la demande CSR et stockez-la dans une variable, car vous en aurez besoin lors d'une étape ultérieure.

   ```
   base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
   ")
   ```

4. Pour créer un fichier nommé `mycsr.yaml`, exécutez la commande suivante. Dans l'exemple suivant, `beta.eks.amazonaws.com/app-serving` est le `signerName`.

   ```
   cat >mycsr.yaml <<EOF
   apiVersion: certificates.k8s.io/v1
   ```

```
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. Envoyez la CSR.

```
kubectl apply -f mycsr.yaml
```

6. Approuvez le certificat de service.

```
kubectl certificate approve myserver
```

7. Vérifiez que le certificat a été émis.

```
kubectl get csr myserver
```

L'exemple qui suit illustre un résultat.

```
NAME       AGE     SIGNERNAME                            REQUESTOR          CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. Exportez le certificat émis.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d >
 myserver.crt
```

Comprendre les rôles et les utilisateurs RBAC créés par Amazon EKS

Lorsque vous créez un cluster Kubernetes, plusieurs identités Kubernetes par défaut sont créées sur ce cluster pour le bon fonctionnement de Kubernetes. Amazon EKS crée des identités Kubernetes pour chacun de ses composants par défaut. Les identités fournissent un contrôle d'autorisation basé

sur les rôles (RBAC) Kubernetes pour les composants du cluster. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Lorsque vous installez des modules complémentaires facultatifs sur votre cluster, des identités Kubernetes supplémentaires peuvent être ajoutées à votre cluster. Pour plus d'informations sur les identités non abordées dans cette rubrique, consultez la documentation du module complémentaire.

Vous pouvez consulter la liste des identités Kubernetes créées par Amazon EKS sur votre cluster à l'aide de l'outil de ligne de `kubectl` commande AWS Management Console or. Toutes les identités des utilisateurs apparaissent dans les journaux `kube` d'audit mis à votre disposition via Amazon CloudWatch.

AWS Management Console

Prérequis

Le principal IAM que vous utilisez doit disposer des autorisations décrites dans la section Autorisations requises.

Pour consulter les identités créées par Amazon EKS à l'aide du AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans la liste Clusters, sélectionnez le cluster qui contient les identités que vous souhaitez afficher.
3. Sélectionnez l'onglet Ressources.
4. Sous Resource types (Types de ressources), sélectionnez Authorization (Autorisation).
5. Choisissez, ClusterRolesClusterRoleBindings, Rôles ou RoleBindings. Toutes les ressources précédées de eks sont créées par Amazon EKS. Les ressources d'identité supplémentaires créées par Amazon EKS sont les suivantes :
   - Le ClusterRoleet ClusterRoleBindingnommé aws-node. Les ressources aws-node prennent en charge le plug-in Amazon VPC CNI pour Kubernetes, qu'Amazon EKS installe sur tous les clusters.
   - Un ClusterRolenom vpc-resource-controller-roleet un ClusterRoleBindingnom vpc-resource-controller-rolebinding. Ces ressources prennent en charge le contrôleur de ressources Amazon VPC, qu'Amazon EKS installe sur tous les clusters.

   Outre les ressources que vous voyez dans la console, les identités utilisateur spéciales suivantes existent sur votre cluster, bien qu'elles ne soient pas visibles dans la configuration du cluster :
   - **`eks:cluster-bootstrap`**— Utilisé pour les `kubectl` opérations lors du bootstrap du cluster.

- **`eks:support-engineer`**— Utilisé pour les opérations de gestion de clusters.

6. Choisissez une ressource spécifique pour afficher les détails la concernant. Par défaut, les informations s'affichent en mode structuré. Dans le coin supérieur droit de la page de détails, vous pouvez sélectionner Raw view (Vue brute) pour afficher toutes les informations relative à la ressource.

Kubectl

Prérequis

L'entité que vous utilisez (AWS Identity and Access Management (IAM) ou OpenID Connect (OIDC)) pour répertorier les ressources Kubernetes du cluster doit être authentifiée par IAM ou par votre fournisseur d'identité OIDC. L'entité doit être autorisée à utiliser les Kubernetes `get` et les `list` verbes pour les `Role` `ClusterRoleBinding` ressources de votre cluster avec lesquelles vous souhaitez que l'entité fonctionne. `ClusterRole` `RoleBinding` Pour plus d'informations sur la façon dont vous pouvez autoriser votre cluster à accéder aux entités IAM, consultez the section called "Accès à l'API Kubernetes". Pour plus d'informations sur l'octroi à des entités authentifiées par votre propre fournisseur OIDC d'un accès à votre cluster, consultez. the section called "Lien vers le fournisseur OIDC"

Pour consulter les identités créées par Amazon EKS à l'aide de **`kubectl`**

Exécutez la commande correspondant au type de ressource que vous souhaitez consulter. Toutes les ressources renvoyées qui sont précédées de eks sont créées par Amazon EKS. Outre les ressources renvoyées dans le résultat des commandes, les identités utilisateur spéciales suivantes existent sur votre cluster, bien qu'elles ne soient pas visibles dans la configuration du cluster :

- **`eks:cluster-bootstrap`**— Utilisé pour les `kubectl` opérations lors du bootstrap du cluster.
- **`eks:support-engineer`**— Utilisé pour les opérations de gestion de clusters.

ClusterRoles— `ClusterRoles` sont limités à votre cluster, de sorte que toute autorisation accordée à un rôle s'applique aux ressources de n'importe quel espace de noms Kubernetes du cluster.

La commande suivante renvoie tous les Kubernetes créés par Amazon EKS `ClusterRoles` sur votre cluster.

```
kubectl get clusterroles | grep eks
```

Outre les `ClusterRoles` renvoyés dans la sortie, les `ClusterRoles` suivants sont disponibles.

- **`aws-node`**— Cela `ClusterRole` prend en charge le plug-in Amazon VPC CNI pour Kubernetes, qu'Amazon EKS installe sur tous les clusters.
- **`vpc-resource-controller-role`**— Cela `ClusterRole` prend en charge le contrôleur de ressources Amazon VPC, qu'Amazon EKS installe sur tous les clusters.

Pour voir la spécification de a`ClusterRole`, remplacez *`eks:k8s-metrics`* la commande suivante par un `ClusterRole` renvoyé dans la sortie de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`ClusterRole`.

```
kubectl describe clusterrole eks:k8s-metrics
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings— `ClusterRoleBindings` sont limités à votre cluster.

La commande suivante renvoie tous les Kubernetes créés par Amazon EKS `ClusterRoleBindings` sur votre cluster.

```
kubectl get clusterrolebindings | grep eks
```

En plus des `ClusterRoleBindings` renvoyés dans la sortie, les `ClusterRoleBindings` suivants sont disponibles.

- **`aws-node`**— Cela `ClusterRoleBinding` prend en charge le plug-in Amazon VPC CNI pour Kubernetes, qu'Amazon EKS installe sur tous les clusters.

- **`vpc-resource-controller-rolebinding`**— Cela `ClusterRoleBinding` prend en charge le contrôleur de ressources Amazon VPC, qu'Amazon EKS installe sur tous les clusters.

Pour voir la spécification de a`ClusterRoleBinding`, remplacez *`eks:k8s-metrics`* la commande suivante par un `ClusterRoleBinding` renvoyé dans la sortie de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`ClusterRoleBinding`.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Rôles : `Roles` ils sont limités à un espace de noms Kubernetes. Tous les `Roles` créés par Amazon EKS sont limités à l'espace de noms `kube-system`.

La commande suivante renvoie tous les Kubernetes créés par Amazon EKS `Roles` sur votre cluster.

```
kubectl get roles -n kube-system | grep eks
```

Pour voir la spécification de a`Role`, remplacez *`eks:k8s-metrics`* la commande suivante par le nom de a `Role` renvoyé dans le résultat de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`Role`.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
```

```
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names             Verbs
  ---------         -----------------  --------------             -----
  daemonsets.apps   []                 [aws-node]                 [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings— `RoleBindings` sont limités à un espace de noms Kubernetes. Tous les `RoleBindings` créés par Amazon EKS sont limités à l'espace de noms `kube-system`.

La commande suivante renvoie tous les Kubernetes créés par Amazon EKS `RoleBindings` sur votre cluster.

```
kubectl get rolebindings -n kube-system | grep eks
```

Pour voir la spécification de a`RoleBinding`, remplacez *`eks:k8s-metrics`* la commande suivante par un `RoleBinding` renvoyé dans la sortie de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`RoleBinding`.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Chiffrez les secrets Kubernetes avec KMS sur les clusters existants

> **Important**
>
> Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou inférieure. Si vous utilisez Kubernetes version 1.28 ou supérieure, vos secrets Kubernetes

sont protégés par un chiffrement d'enveloppe par défaut. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

Si vous activez le chiffrement des secrets, les secrets Kubernetes sont chiffrés à l'aide de la clé AWS KMS que vous sélectionnez. La clé KMS doit répondre aux conditions suivantes :

- Symétrique
- Peut chiffrer et déchiffrer des données
- Créé dans la même AWS région que le cluster
- Si la clé KMS a été créée dans un autre compte, le principal IAM doit avoir accès à la clé KMS.

Pour plus d'informations, consultez la section Autoriser les principaux IAM d'autres comptes à utiliser une clé KMS dans le Guide du développeur du service de gestion des AWS clés.

> **Warning**
>
> Vous ne pouvez pas désactiver le chiffrement des secrets après l'avoir activé. Cette action est irréversible.

eksctl

Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou inférieure. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

Vous pouvez activer le chiffrement de deux manières :

- Ajoutez le chiffrement à votre cluster à l'aide d'une seule commande.

  Pour re-chiffrer automatiquement vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key
```

Pour vous désabonner du chiffrement automatique de vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- Ajoutez le chiffrement à votre cluster à l'aide d'un fichier `kms-cluster.yaml`.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws: kms:region-code:account:key/key
```

Pour re-chiffrer automatiquement vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

Pour vous désabonner du chiffrement automatique de vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

AWS Management Console

a. Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou inférieure. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

b. Ouvrez la console Amazon EKS.

c. Choisissez le cluster auquel vous souhaitez ajouter le chiffrement KMS.

d. Cliquez sur l'onglet Présentation (cette option est sélectionnée par défaut).

e. Faites défiler la page jusqu'à Secrets encryption (Chiffrement des secrets) et sélectionnez Enable (Activer).

f. Sélectionnez une clé dans la liste déroulant et appuyez sur le bouton Enable (Activer). Si aucune clé n'est répertoriée, vous devez d'abord en créer une. Pour plus d'informations, consultez Création de clés.

g. Appuyez sur le bouton Confirm (Confirmer) pour utiliser la touche choisie.

AWS CLI

a. Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou inférieure. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

b. Associez la configuration de chiffrement des secrets à votre cluster à l'aide de la commande AWS CLI suivante. Remplacez les *example values* par vos propres valeurs.

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws: kms:region-code:account:key/key"}}]'
```

L'exemple qui suit illustre un résultat.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

c. Vous pouvez contrôler l'état de votre mise à jour de chiffrement à l'aide de la commande suivante. Utilisez le `cluster name` et le `update ID` spécifiques qui ont été renvoyés dans la sortie précédente. Lorsqu'un état `Successful` s'affiche, la mise à jour est terminée.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

L'exemple qui suit illustre un résultat.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

d. Pour vérifier que le chiffrement est activé dans votre cluster, exécutez la commande `describe-cluster`. La réponse contient une chaîne `EncryptionConfig`.

```
aws eks describe-cluster --region region-code --name my-cluster
```

Une fois que vous avez activé le chiffrement sur votre cluster, vous devez chiffrer tous les secrets existants avec la nouvelle clé :

> **Note**
>
> Si vous utilisez `eksctl`, l'exécution de la commande suivante n'est nécessaire que si vous vous désabonnez du chiffrement automatique de vos secrets.

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

> **Warning**
>
> Si vous activez le chiffrement secret pour un cluster existant et que la clé KMS que vous utilisez est supprimée, il n'y a aucun moyen de récupérer le cluster. Si vous supprimez la clé KMS, vous mettez définitivement le cluster dans un état dégradé. Pour plus d'informations, consultez la section Suppression de clés AWS KMS.

> **Note**
>
> Par défaut, la `create-key` commande crée une clé KMS de chiffrement symétrique avec une politique de clé qui donne à l'administrateur racine du compte un accès aux actions et aux ressources AWS KMS. Si vous souhaitez limiter les autorisations, assurez-vous que les actions `kms:DescribeKey` et `kms:CreateGrant` sont autorisées sur la stratégie pour le principal qui appellera l'API `create-cluster`.
> Pour les clusters utilisant le chiffrement d'enveloppe KMS, des autorisations `kms:CreateGrant` sont requises. La condition n'`kms:GrantIsForAWSResource`est pas prise en charge pour l' CreateCluster action et ne doit pas être utilisée dans les politiques KMS pour contrôler `kms:CreateGrant` les autorisations accordées aux utilisateurs CreateCluster.

Utiliser AWS les secrets de Secrets Manager avec les pods Amazon EKS

Pour afficher les secrets de Secrets Manager et les paramètres de Parameter Store sous forme de fichiers montés dans des pods Amazon EKS, vous pouvez utiliser le fournisseur de AWS secrets et de configuration (ASCP) pour le pilote CSI Kubernetes Secrets Store.

Avec l'ASCP, vous pouvez stocker et gérer vos secrets dans Secrets Manager, puis les récupérer via vos applications exécutées sur Amazon EKS. Vous pouvez utiliser les rôles et les politiques IAM pour limiter l'accès à vos secrets à des pods Kubernetes spécifiques d'un cluster. L'ASCP récupère l'identité du Pod et échange l'identité contre un rôle IAM. L'ASCP assume le rôle IAM du Pod, puis il peut récupérer les secrets autorisés pour ce rôle auprès de Secrets Manager.

Si vous utilisez la rotation automatique de Secrets Manager pour vos secrets, vous pouvez également utiliser la fonction de réconciliation de rotation du pilote CSI de Secrets Store pour vous assurer que vous récupérez le dernier secret de Secrets Manager.

> **Note**
>
> AWS Les groupes de nœuds Fargate (Fargate) ne sont pas pris en charge.

Pour plus d'informations, consultez la section Utilisation des secrets de Secrets Manager dans Amazon EKS dans le guide de l'utilisateur de AWS Secrets Manager.

Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) fournit un chiffrement d'enveloppe par défaut pour toutes les données d'API Kubernetes dans les clusters EKS exécutant Kubernetes version 1.28 ou ultérieure.

Le chiffrement des enveloppes protège les données que vous stockez avec le serveur d'API Kubernetes. Par exemple, le chiffrement d'enveloppe s'applique à la configuration de votre cluster Kubernetes, par exemple. `ConfigMaps` Le chiffrement des enveloppes ne s'applique pas aux données des nœuds ou des volumes EBS. EKS prenait auparavant en charge le chiffrement des secrets Kubernetes, et ce chiffrement d'enveloppe s'étend désormais à toutes les données de l'API Kubernetes.

Cela fournit une expérience gérée par défaut qui s'implémente defense-in-depth pour vos applications Kubernetes et ne nécessite aucune action de votre part.

Amazon EKS utilise le service de gestion des AWS clés (KMS) avec le fournisseur Kubernetes KMS v2 pour cette couche de sécurité supplémentaire avec une clé appartenant à Amazon Web Services et la possibilité pour vous d'apporter votre propre clé gérée par le client (CMK) auprès de KMS. AWS

Comprendre le chiffrement des enveloppes

Le chiffrement d'enveloppe consiste à chiffrer des données en texte brut à l'aide d'une clé de chiffrement des données (DEK) avant leur envoi à la banque de données (etcd), puis à chiffrer la DEK avec une clé KMS racine stockée dans un système KMS (KMS) distant géré de manière centralisée (KMS).AWS Il s'agit d'une defense-in-depth stratégie car elle protège les données à l'aide d'une clé de chiffrement (DEK), puis ajoute une autre couche de sécurité en protégeant cette DEK avec une clé de chiffrement séparée et stockée de manière sécurisée appelée clé de chiffrement (KEK).

Comment Amazon EKS active le chiffrement des enveloppes par défaut avec KMS v2 et AWS KMS

Amazon EKS utilise KMS v2 pour implémenter le chiffrement d'enveloppe par défaut pour toutes les données d'API dans le plan de contrôle Kubernetes géré avant qu'elles ne soient conservées dans la base de données etcd. Au démarrage, le serveur API du cluster génère une clé de chiffrement des données (DEK) à partir d'une graine secrète combinée à des données générées de manière aléatoire. Au démarrage également, le serveur d'API appelle le plug-in KMS pour chiffrer le seed DEK à l'aide d'une clé de chiffrement par clé distante (KEK) fournie par AWS KMS. Il s'agit d'un appel ponctuel exécuté au démarrage du serveur API et lors de la rotation du KEK. Le serveur API met ensuite en cache la graine DEK chiffrée. Ensuite, le serveur API utilise la graine DEK mise en cache pour générer d'autres données à usage unique DEKs basées sur une fonction de dérivation de clés (KDF). Chacun de ces éléments générés n' DEKs est ensuite utilisé qu'une seule fois pour chiffrer une seule ressource Kubernetes avant qu'elle ne soit stockée dans etcd. Grâce à l'utilisation d'une graine DEK cryptée en cache dans KMS v2, le processus de chiffrement des ressources Kubernetes sur le serveur d'API est à la fois plus performant et plus rentable.

Par défaut, ce KEK appartient à AWS KMS AWS, mais vous pouvez éventuellement apporter le vôtre.

Le schéma ci-dessous décrit la génération et le chiffrement d'un DEK au démarrage du serveur d'API.

Le schéma de haut niveau ci-dessous décrit le chiffrement d'une ressource Kubernetes avant son stockage dans etcd.

Questions fréquentes (FAQ)

Comment le chiffrement d'enveloppe par défaut améliore-t-il le niveau de sécurité de mon cluster EKS ?

Cette fonctionnalité réduit la surface et la durée pendant lesquelles les métadonnées et le contenu client ne sont pas chiffrés. Avec le chiffrement d'enveloppe par défaut, les métadonnées et le contenu client ne sont que temporairement déchiffrés dans la mémoire du serveur kube-apiserver avant d'être stockés dans etcd. La mémoire du kube-apiserver est sécurisée via le système Nitro. Amazon EKS utilise uniquement des EC2 instances basées sur Nitro pour le plan de contrôle Kubernetes géré. Ces instances ont des conceptions de contrôle de sécurité qui empêchent tout système ou toute personne d'accéder à leur mémoire.

Quelle version de Kubernetes dois-je exécuter pour bénéficier de cette fonctionnalité ?

Pour que le chiffrement d'enveloppe par défaut soit activé, votre cluster Amazon EKS doit exécuter Kubernetes version 1.28 ou ultérieure.

Mes données sont-elles toujours sécurisées si j'utilise une version du cluster Kubernetes qui ne prend pas en charge cette fonctionnalité ?

Oui. Chez AWS, la sécurité est notre priorité absolue. Nous basons toute notre transformation numérique et notre innovation sur les meilleures pratiques opérationnelles en matière de sécurité, et nous restons déterminés à relever cette barre.

Toutes les données stockées dans l'etcd sont cryptées au niveau du disque pour chaque cluster EKS, quelle que soit la version de Kubernetes exécutée. EKS utilise des clés racines qui génèrent des clés de chiffrement de volume qui sont gérées par le service EKS. En outre, chaque cluster Amazon EKS est exécuté dans un VPC isolé à l'aide de machines virtuelles spécifiques au cluster. Grâce à cette architecture et à nos pratiques en matière de sécurité opérationnelle, Amazon EKS a obtenu plusieurs notations de conformité et normes, notamment l'éligibilité aux normes SOC 1,2,3, PCI-DSS, ISO et HIPAA. Ces niveaux et normes de conformité sont maintenus pour tous les clusters EKS avec ou sans chiffrement d'enveloppe par défaut.

Comment fonctionne le chiffrement des enveloppes dans Amazon EKS ?

Au démarrage, le serveur API du cluster génère une clé de chiffrement des données (DEK) à partir d'une graine secrète combinée à des données générées de manière aléatoire. Au démarrage également, le serveur d'API appelle le plug-in KMS pour chiffrer le DEK à l'aide d'une clé de

chiffrement à distance (KEK) fournie par AWS KMS. Il s'agit d'un appel ponctuel exécuté au démarrage du serveur API et lors de la rotation du KEK. Le serveur API met ensuite en cache la graine DEK chiffrée. Ensuite, le serveur API utilise la graine DEK mise en cache pour générer d'autres données à usage unique DEKs basées sur une fonction de dérivation de clés (KDF). Chacun de ces éléments générés n' DEKs est ensuite utilisé qu'une seule fois pour chiffrer une seule ressource Kubernetes avant qu'elle ne soit stockée dans etcd.

Il est important de noter que des appels supplémentaires sont effectués depuis le serveur API pour vérifier le bon fonctionnement et le fonctionnement normal de l'intégration AWS KMS. Ces bilans de santé supplémentaires sont visibles dans votre AWS CloudTrail.

Dois-je faire quelque chose ou modifier des autorisations pour que cette fonctionnalité fonctionne dans mon cluster EKS ?

Non, vous n'avez rien à faire. Le chiffrement des enveloppes dans Amazon EKS est désormais une configuration par défaut activée dans tous les clusters exécutant Kubernetes version 1.28 ou ultérieure. L'intégration AWS KMS est établie par le serveur d'API Kubernetes géré par. AWS Cela signifie que vous n'avez pas besoin de configurer d'autorisations pour commencer à utiliser le chiffrement KMS pour votre cluster.

Comment savoir si le chiffrement d'enveloppe par défaut est activé sur mon cluster ?

Si vous effectuez la migration pour utiliser votre propre clé CMK, vous verrez l'ARN de la clé KMS associée à votre cluster. En outre, vous pouvez consulter les journaux AWS CloudTrail d'événements associés à l'utilisation de la clé CMK de votre cluster.

Si votre cluster utilise une clé AWS détenue, celle-ci sera détaillée dans la console EKS (à l'exception de l'ARN de la clé).

Puis-je AWS accéder à la clé AWS détenue utilisée pour le chiffrement des enveloppes par défaut dans Amazon EKS ?

Non. AWS dispose de contrôles de sécurité stricts dans Amazon EKS qui empêchent toute personne d'accéder aux clés de chiffrement en texte brut utilisées pour sécuriser les données de la base de données etcd. Ces mesures de sécurité sont également appliquées à la clé KMS AWS détenue.

Le chiffrement d'enveloppe par défaut est-il activé dans mon cluster EKS existant ?

Si vous utilisez un cluster Amazon EKS avec Kubernetes version 1.28 ou supérieure, le chiffrement d'enveloppe de toutes les données de l'API Kubernetes est activé. Pour les clusters existants,

Amazon EKS utilise le `eks:kms-storage-migrator` RBAC ClusterRole pour faire migrer les données qui n'étaient pas auparavant chiffrées par enveloppe dans etcd vers ce nouvel état de chiffrement.

Qu'est-ce que cela signifie si j'ai déjà activé le chiffrement d'enveloppe pour les secrets dans mon cluster EKS ?

Si vous possédez déjà une clé gérée par le client (CMK) dans KMS qui a été utilisée pour chiffrer vos secrets Kubernetes, cette même clé sera utilisée comme clé KEK pour le chiffrement de l'enveloppe de tous les types de données d'API Kubernetes de votre cluster.

L'exécution d'un cluster EKS avec le chiffrement d'enveloppe par défaut entraîne-t-elle des coûts supplémentaires ?

Aucun coût supplémentaire n'est associé au plan de contrôle Kubernetes géré si vous utilisez une clé appartenant à Amazon Web Services pour le chiffrement de l'enveloppe par défaut. Par défaut, chaque cluster EKS exécutant Kubernetes version 1.28 ou ultérieure utilise une clé appartenant à Amazon Web Service. Toutefois, si vous utilisez votre propre clé AWS KMS, la tarification KMS normale s'appliquera.

Combien coûte l'utilisation de ma propre clé AWS KMS pour chiffrer les données de l'API Kubernetes dans mon cluster ?

Vous payez 1$ par mois pour stocker toute clé personnalisée que vous créez ou importez dans KMS. KMS facture les demandes de chiffrement et de déchiffrement. Il existe un niveau gratuit de 20 000 demandes par mois et par compte et vous payez 0,03$ par 10 000 demandes de plus que le niveau gratuit par mois. Cela s'applique à toutes les utilisations de KMS pour un compte, de sorte que le coût d'utilisation de votre propre clé AWS KMS sur votre cluster sera affecté par l'utilisation de cette clé sur d'autres clusters ou AWS ressources de votre compte.

Mes frais KMS seront-ils plus élevés maintenant que ma clé gérée par le client (CMK) est utilisée pour crypter toutes les données de l'API Kubernetes, et pas seulement les secrets ?

Non. Notre mise en œuvre avec KMS v2 réduit considérablement le nombre d'appels passés à AWS KMS. Cela réduira à son tour les coûts associés à votre CMK, quelles que soient les données Kubernetes supplémentaires chiffrées ou déchiffrées dans votre cluster EKS.

Comme indiqué ci-dessus, la graine DEK générée utilisée pour le chiffrement des ressources Kubernetes est stockée localement dans le cache du serveur d'API Kubernetes après avoir été chiffrée avec le KEK distant. Si la graine DEK chiffrée ne se trouve pas dans le cache du serveur

API, le serveur API appellera AWS KMS pour chiffrer la graine DEK. Le serveur d'API met ensuite en cache la graine DEK chiffrée pour une utilisation future dans le cluster sans appeler KMS. De même, pour les demandes de déchiffrement, le serveur API appellera AWS KMS pour la première demande de déchiffrement, après quoi la graine DEK déchiffrée sera mise en cache et utilisée pour les futures opérations de déchiffrement.

Pour plus d'informations, consultez KEP-3299 : Améliorations de KMS v2 dans les améliorations de Kubernetes sur. GitHub

Puis-je utiliser la même clé CMK pour plusieurs clusters Amazon EKS ?

Oui. Pour réutiliser une clé, vous pouvez la lier à un cluster de la même région en associant l'ARN au cluster lors de sa création. Toutefois, si vous utilisez la même clé CMK pour plusieurs clusters EKS, vous devez mettre en place les mesures nécessaires pour empêcher la désactivation arbitraire de la clé CMK. Sinon, une clé CMK désactivée associée à plusieurs clusters EKS aura un impact plus large sur les clusters en fonction de cette clé.

Qu'arrive-t-il à mon cluster EKS si ma clé CMK devient indisponible après l'activation du chiffrement d'enveloppe par défaut ?

Si vous désactivez une clé KMS, elle ne peut être utilisée dans aucune opération cryptographique. Sans accès à une clé CMK existante, le serveur API ne sera pas en mesure de chiffrer et de conserver les objets Kubernetes nouvellement créés, ni de déchiffrer les objets Kubernetes précédemment chiffrés stockés dans etcd. Si la clé CMK est désactivée, le cluster sera immédiatement placé dans un état malsain/dégradé, auquel cas nous ne serons pas en mesure de respecter notre engagement de service tant que vous n'aurez pas réactivé la clé CMK associée.

Lorsqu'une clé CMK est désactivée, vous recevez des notifications concernant la détérioration de l'état de santé de votre cluster EKS et la nécessité de réactiver votre clé CMK dans les 30 jours suivant sa désactivation afin de garantir la restauration réussie des ressources de votre plan de contrôle Kubernetes.

Comment protéger mon cluster EKS de l'impact d'une clé CMK désactivée/supprimée ?

Pour protéger vos clusters EKS contre un tel événement, vos administrateurs clés doivent gérer l'accès aux opérations clés KMS en utilisant des politiques IAM avec le principe du moindre privilège afin de réduire le risque de désactivation ou de suppression arbitraire des clés associées aux clusters EKS. De plus, vous pouvez configurer une CloudWatch alarme pour être informé de l'état de votre CMK.

Mon cluster EKS sera-t-il restauré si je réactive le CMK ?

Pour garantir une restauration réussie de votre cluster EKS, nous vous recommandons vivement de réactiver votre clé CMK dans les 30 premiers jours suivant sa désactivation. Cependant, le succès de la restauration de votre cluster EKS dépendra également de la question de savoir s'il subit ou non des modifications perturbatrices de l'API en raison d'une mise à niveau automatique de Kubernetes qui peut avoir lieu alors que le cluster est dans un état malsain ou dégradé.

Pourquoi mon cluster EKS est-il placé dans un état malsain/dégradé après avoir désactivé le CMK ?

Le serveur API du plan de contrôle EKS utilise une clé DEK cryptée et mise en cache dans la mémoire du serveur API pour chiffrer tous les objets lors des opérations de création/mise à jour avant qu'ils ne soient stockés dans etcd. Lorsqu'un objet existant est récupéré depuis etcd, le serveur API utilise la même clé DEK mise en cache et déchiffre l'objet de ressource Kubernetes. Si vous désactivez le CMK, le serveur d'API ne subira aucun impact immédiat en raison de la clé DEK mise en cache dans la mémoire du serveur d'API. Toutefois, lorsque l'instance du serveur d'API est redémarrée, elle ne possède pas de DEK en cache et devra appeler AWS KMS pour les opérations de chiffrement et de déchiffrement. Sans clé CMK, ce processus échouera avec un code d'erreur KMS_KEY_DISABLED, empêchant le serveur d'API de démarrer correctement.

Qu'arrive-t-il à mon cluster EKS si je supprime mon CMK ?

La suppression de la clé CMK associée à votre cluster EKS dégradera son état de santé de manière irréversible. Sans la clé CMK de votre cluster, le serveur API ne sera plus en mesure de chiffrer et de conserver les nouveaux objets Kubernetes, ni de déchiffrer les objets Kubernetes précédemment chiffrés stockés dans la base de données etcd. Vous ne devez procéder à la suppression d'une clé CMK pour votre cluster EKS que lorsque vous êtes certain de ne plus avoir besoin de l'utiliser.

Notez que si la clé CMK n'est pas trouvée (KMS_KEY_NOT_FOUND) ou si les autorisations associées à la clé CMK associées à votre cluster sont révoquées (KMS_GRANT_REVOKED), votre cluster ne sera pas récupérable. Pour plus d'informations sur l'état du cluster et les codes d'erreur, voir État du cluster FAQs et codes d'erreur avec chemins de résolution.

Serai-je toujours facturé pour un cluster EKS dégradé/défectueux parce que j'ai désactivé ou supprimé ma clé CMK ?

Oui. Bien que le plan de contrôle EKS ne soit pas utilisable en cas de désactivation d'une clé CMK, les ressources d'infrastructure dédiées allouées au cluster EKS AWS continueront d'être utilisées jusqu'à ce qu'il soit supprimé par le client. En outre, notre engagement de service ne s'appliquera

pas dans de telles circonstances, car il s'agira d'une action ou d'une inaction volontaire du client qui empêchera la santé et le fonctionnement normaux de votre cluster EKS.

Mon cluster EKS peut-il être automatiquement mis à niveau lorsqu'il est dans un état malsain ou dégradé en raison d'une clé CMK désactivée ?

Oui. Toutefois, si votre cluster possède une clé CMK désactivée, vous aurez 30 jours pour la réactiver. Au cours de cette période de 30 jours, votre cluster Kubernetes ne sera pas automatiquement mis à niveau. Toutefois, si cette période expire et que vous n'avez pas réactivé le CMK, le cluster sera automatiquement mis à niveau vers la version suivante (n+1) compatible avec le support standard, en suivant le cycle de vie des versions de Kubernetes dans EKS.

Nous vous recommandons vivement de réactiver rapidement une clé CMK désactivée lorsque vous prenez connaissance d'un cluster concerné. Il est important de noter que même si EKS met automatiquement à niveau les clusters concernés, rien ne garantit qu'ils seront restaurés avec succès, en particulier si le cluster fait l'objet de plusieurs mises à niveau automatiques, car cela peut inclure des modifications de l'API Kubernetes et un comportement inattendu dans le processus de démarrage du serveur d'API.

Puis-je utiliser un alias de clé KMS ?

Oui. Amazon EKS prend en charge l'utilisation d'alias de clé KMS. Un alias est un nom convivial pour une clé KMS Amazon Web Service. Par exemple, un alias vous permet de faire référence à une clé KMS sous le nom de « my-key » au lieu de **`1234abcd-12ab-34cd-56ef-1234567890ab`**.

Puis-je toujours sauvegarder et restaurer les ressources de mon cluster à l'aide de ma propre solution de sauvegarde Kubernetes ?

Oui. Vous pouvez utiliser une solution de sauvegarde Kubernetes (telle que Velero) pour la reprise après sinistre du cluster Kubernetes, la migration des données et la protection des données. Si vous exécutez une solution de sauvegarde Kubernetes qui accède aux ressources du cluster via le serveur d'API, toutes les données récupérées par l'application seront déchiffrées avant d'atteindre le client. Cela vous permettra de récupérer les ressources du cluster dans un autre cluster Kubernetes.

Considérations relatives à la sécurité pour le mode automatique d'Amazon EKS

Cette rubrique décrit l'architecture de sécurité, les contrôles et les meilleures pratiques pour Amazon EKS Auto Mode. À mesure que les entreprises déploient des applications conteneurisées à grande

échelle, le maintien d'une posture de sécurité solide devient de plus en plus complexe. Le mode automatique d'EKS met en œuvre des contrôles de sécurité automatisés et s'intègre aux services de AWS sécurité pour vous aider à protéger l'infrastructure, les charges de travail et les données de votre cluster. Grâce à des fonctionnalités de sécurité intégrées telles que la gestion renforcée du cycle de vie des nœuds et le déploiement automatique des correctifs, le mode automatique d'EKS vous aide à maintenir les meilleures pratiques en matière de sécurité tout en réduisant les frais d'exploitation.

Avant de poursuivre cette rubrique, assurez-vous que vous connaissez les concepts de base du mode automatique EKS et que vous avez examiné les conditions préalables à l'activation du mode automatique EKS sur vos clusters. Pour obtenir des informations générales sur la sécurité d'Amazon EKS, consultezSécurité.

Le mode automatique d'Amazon EKS s'appuie sur les bases de sécurité existantes d'Amazon EKS tout en introduisant des contrôles de sécurité automatisés supplémentaires pour les instances EC2 gérées.

Sécurité et authentification des API

Le mode automatique Amazon EKS utilise les mécanismes de sécurité de la AWS plateforme pour sécuriser et authentifier les appels vers l'API Amazon EKS.

- L'accès à l'API Kubernetes est sécurisé par le biais d'entrées d'accès EKS, qui s'intègrent aux AWS identités IAM.
  - Pour de plus amples informations, veuillez consulter the section called “Accorder des autorisations”.
- Les clients peuvent mettre en œuvre un contrôle d'accès précis au point de terminaison de l'API Kubernetes en configurant les entrées d'accès EKS.

Sécurité du réseau

Le mode automatique Amazon EKS prend en charge plusieurs niveaux de sécurité réseau :

- Intégration VPC
  - Fonctionne au sein de votre Amazon Virtual Private Cloud (VPC)
  - Prend en charge les configurations VPC personnalisées et les configurations de sous-réseaux
  - Permet la mise en réseau privée entre les composants du cluster

  - Pour plus d'informations, consultez Gérer les responsabilités en matière de sécurité pour Amazon Virtual Private Cloud
- Politiques du réseau
  - Support natif pour les politiques réseau Kubernetes
  - Possibilité de définir des règles de trafic réseau granulaires
  - Pour plus d'informations, consultez the section called "Politiques Kubernetes".

EC2 sécurité des instances gérées

Amazon EKS Auto Mode gère les instances EC2 gérées avec les contrôles de sécurité suivants :

EC2 sécurité

- EC2 les instances gérées conservent les fonctionnalités de sécurité d'Amazon EC2.
- Pour plus d'informations sur les instances EC2 gérées, consultez la section Sécurité sur Amazon EC2.

Gestion du cycle de vie des instances

EC2 les instances gérées gérées par le mode automatique EKS ont une durée de vie maximale de 21 jours. Le mode automatique d'Amazon EKS met automatiquement fin aux instances dépassant cette durée de vie. Cette limite de cycle de vie permet d'éviter les dérives de configuration et de maintenir le niveau de sécurité.

Protection des données

- Amazon EC2 Instance Storage est crypté, il s'agit d'un stockage directement attaché à l'instance. Pour plus d'informations, consultez la section Protection des données sur Amazon EC2.
- Le mode automatique EKS gère les volumes attachés aux EC2 instances au moment de leur création, y compris les volumes racine et de données. Le mode automatique EKS ne gère pas entièrement les volumes EBS créés à l'aide des fonctionnalités de stockage persistant de Kubernetes.

Gestion des correctifs

- Le mode automatique Amazon EKS applique automatiquement des correctifs aux instances gérées.
- Les correctifs incluent :
  - Mises à jour du système d'exploitation
  - Correctifs de sécurité
  - Composants du mode automatique Amazon EKS

> **Note**
>
> Les clients restent responsables de la sécurisation et de la mise à jour des charges de travail exécutées sur ces instances.

Contrôles d'accès

- L'accès direct à l'instance est restreint :
  - L'accès SSH n'est pas disponible.
  - AWS L'accès au gestionnaire de session (SSM) de Systems Manager n'est pas disponible.
- Les opérations de gestion sont effectuées via l'API Amazon EKS et l'API Kubernetes.

Gestion automatisée des ressources

Amazon EKS Auto Mode ne gère pas entièrement les volumes Amazon Elastic Block Store (Amazon EBS) créés à l'aide des fonctionnalités de stockage persistant de Kubernetes. Le mode automatique EKS ne gère pas non plus les Elastic Load Balancers (ELB). Le mode automatique d'Amazon EKS automatise les tâches de routine associées à ces ressources.

Sécurité du stockage

- AWS vous recommande d'activer le chiffrement pour les volumes EBS fournis par les fonctionnalités de stockage persistant de Kubernetes. Pour de plus amples informations, veuillez consulter the section called “Créez StorageClass”.
- Chiffrement au repos à l'aide de AWS KMS

- Vous pouvez configurer votre AWS compte pour appliquer le chiffrement des nouveaux volumes EBS et des copies instantanées que vous créez. Pour plus d'informations, consultez Activer le chiffrement Amazon EBS par défaut dans le guide de l'utilisateur Amazon EBS.
- Pour plus d'informations, consultez la section Sécurité dans Amazon EBS.

Sécurité de l'équilibreur de charge

- Configuration automatisée des équilibreurs de charge élastiques
- Gestion des certificats SSL/TLS via l'intégration de Certificate Manager AWS
- Automatisation des groupes de sécurité pour le contrôle d'accès aux équilibreurs de charge
- Pour plus d'informations, consultez la section Sécurité dans Elastic Load Balancing.

Bonnes pratiques de sécurité

La section suivante décrit les meilleures pratiques de sécurité pour le mode automatique d'Amazon EKS.

- Passez régulièrement en revue les politiques AWS IAM et les entrées d'accès EKS.
- Mettez en œuvre des modèles d'accès avec le moindre privilège pour les charges de travail.
- Surveillez l'activité du cluster via AWS CloudTrail Amazon CloudWatch. Pour plus d'informations, consultez the section called " AWS CloudTrail" et the section called "Amazon CloudWatch".
- Utilisez AWS Security Hub pour évaluer le niveau de sécurité.
- Mettez en œuvre des normes de sécurité adaptées à vos charges de travail.

Gestion des identités et des accès pour Amazon EKS

AWS Identity and Access Management (IAM) est un AWS service qui permet à un administrateur de contrôler en toute sécurité l'accès aux ressources. AWS Des administrateurs IAM contrôlent les personnes qui peuvent être authentifiées (connectées) et autorisées (disposant d'autorisations) pour utiliser des ressources Amazon EKS. IAM est un AWS service que vous pouvez utiliser sans frais supplémentaires.

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans Amazon EKS.

Utilisateur du service : si vous utilisez le service Amazon EKS pour accomplir votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Vous pourrez avoir besoin d'autorisations supplémentaires si vous utilisez davantage de fonctionnalités Amazon EKS. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans Amazon EKS, consultez the section called "Résolution des problèmes".

Administrateur du service : si vous êtes responsable des ressources Amazon EKS au sein de votre entreprise, vous disposez probablement d'un accès complet à Amazon EKS. C'est à vous de déterminer les fonctionnalités et les ressources Amazon EKS auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec Amazon EKS, consultez the section called "Amazon EKS et IAM".

Administrateur IAM : si vous êtes administrateur IAM, vous souhaiterez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à Amazon EKS. Pour afficher des exemples de politiques basées sur l'identité Amazon EKS que vous pouvez utiliser dans IAM, consultez the section called "Politiques basées sur l'identité".

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur root du AWS compte, en tant qu'utilisateur IAM ou en assumant un rôle IAM.

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS Les utilisateurs de l'IAM Identity Center (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez Comment vous connecter à votre AWS compte dans le Guide de l'utilisateur de AWS connexion.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section Signature des demandes AWS d'API dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez Authentification multifactorielle dans le guide de l'utilisateur d' AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) AWS dans le guide de l'utilisateur IAM.

AWS utilisateur root du compte

Lorsque vous créez un AWS compte, vous commencez par utiliser une seule identité de connexion qui donne un accès complet à tous les AWS services et ressources du compte. Cette identité est appelée utilisateur root du AWS compte et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur root pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez Tâches nécessitant les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de votre AWS compte qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe et des clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons d'effectuer une rotation des clés d'accès. Pour plus d'informations, consultez Rotation régulière des

clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de votre AWS compte dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API ou de AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les ensembles d'autorisations, consultez la section Ensembles d'autorisations dans le guide de l'utilisateur d' AWS IAM Identity Center.
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent

le principal moyen d'accorder l'accès intercompte. Cependant, avec certains AWS services, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

- Accès interservices — Certains AWS services utilisent des fonctionnalités d'autres AWS services. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, une fonction du service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. Le FAS utilise les autorisations du principal appelant un AWS service, associées au AWS service demandeur pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres AWS services ou ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle lié à un service. AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés au service apparaissent dans votre AWS compte et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des requêtes AWS CLI ou AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez

la section Utilisation d'un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le AWS Management Console rôle à partir de la AWS CLI ou de l' AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles dans votre AWS compte. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou AWS des services.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3, AWS WAF et Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations qui en résultent sont l'intersection des politiques basées sur l'identité d'une entité et de ses limites d'autorisations. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée plusieurs AWS comptes détenus par votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités des comptes membres, y compris pour chaque utilisateur root AWS du compte. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de l'utilisateur AWS des Organizations.
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la session obtenue sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de session. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Fonctionnement d'Amazon EKS avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon EKS, vous devez comprendre quelles sont les fonctionnalités IAM qui peuvent être utilisées dans cette situation. Pour obtenir une vue d'ensemble

de la manière dont Amazon EKS et les autres AWS services fonctionnent avec IAM, consultez la section AWS Services compatibles avec IAM dans le guide de l'utilisateur d'IAM.

Rubriques

- Politiques basées sur l'identité Amazon EKS
- Politiques basées sur les ressources Amazon EKS
- Autorisation basée sur des identifications Amazon EKS
- Rôles IAM Amazon EKS

Politiques basées sur l'identité Amazon EKS

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Amazon EKS est compatible avec des actions, des ressources et des clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe certaines exceptions, telles que les actions avec autorisation uniquement pour lesquelles aucune opération d'API correspondante n'est associée. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions de politique dans Amazon EKS utilisent le préfixe suivant avant l'action : `eks:`. Par exemple, pour accorder à une personne l'autorisation d'obtenir des informations descriptives sur un cluster Amazon EKS, incluez l'action `DescribeCluster` dans sa politique. Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": ["eks:action1", "eks:action2"]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "eks:Describe*"
```

Pour afficher la liste des actions Amazon EKS, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne prennent pas en charge les autorisations au niveau des ressources, telles que les opérations de listage, utilisez un caractère générique (*) pour indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

La ressource de cluster Amazon EKS intègre l'ARN suivant.

```
 arn:aws: eks:region-code:account-id:cluster/cluster-name
```

Pour plus d'informations sur le format de ARNs, consultez Amazon resource names (ARNs) et espaces de noms AWS de services Amazon.

Par exemple, pour spécifier le cluster dont le nom figure *`my-cluster`* dans votre instruction, utilisez l'ARN suivant :

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/my-cluster"
```

Pour spécifier tous les clusters appartenant à un compte et à une AWS région spécifiques, utilisez le caractère générique (*) :

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/*"
```

Certaines actions Amazon EKS, telles que celles relatives à la création de ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

```
"Resource": "*"
```

Pour consulter la liste des types de ressources Amazon EKS et leurs caractéristiques ARNs, consultez la section Ressources définies par Amazon Elastic Kubernetes Service dans le Service Authorization Reference. Pour connaître les actions avec lesquelles vous pouvez spécifier l'ARN de chaque ressource, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

Clés de condition

Amazon EKS définit son propre ensemble de clés de condition et est également compatible avec l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, consultez la section Clés contextuelles de condition AWS globale dans le guide de l'utilisateur IAM.

Vous pouvez définir des clés de condition lors de l'association d'un fournisseur OpenID Connect à votre cluster. Pour de plus amples informations, veuillez consulter the section called “Exemple de politique IAM”.

Toutes les EC2 actions Amazon prennent en charge les clés de `ec2:Region` condition `aws:RequestedRegion` et. Pour plus d'informations, voir Exemple : restriction de l'accès à une AWS région spécifique.

Pour obtenir la liste des clés de condition Amazon EKS, consultez la rubrique Conditions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

Exemples

Pour voir des exemples de politiques Amazon EKS basées sur l'identité, consultez the section called "Politiques basées sur l'identité".

Lorsque vous créez un cluster Amazon EKS, le principal IAM qui crée le cluster reçoit automatiquement des `system:masters` autorisations dans la configuration du contrôle d'accès basé sur les rôles (RBAC) du cluster dans le plan de contrôle Amazon EKS. Ce principal n'apparaît dans aucune configuration visible. Assurez-vous donc de savoir quel principal a créé le cluster à l'origine. Pour autoriser des principaux IAM supplémentaires à interagir avec votre cluster, modifiez-le `aws-auth ConfigMap` dans Kubernetes et créez un Kubernetes `rolebinding` ou `clusterrolebinding` avec le nom d'un Kubernetes que vous spécifiez dans le. `group` `aws-auth ConfigMap`

Pour plus d'informations sur l'utilisation du ConfigMap, consultezthe section called "Accès à l'API Kubernetes".

Politiques basées sur les ressources Amazon EKS

Amazon EKS ne prend pas en charge les politiques basées sur les ressources.

Autorisation basée sur des identifications Amazon EKS

Vous pouvez attacher des identifications aux ressources Amazon EKS ou transmettre des identifications dans une demande à Amazon EKS. Pour contrôler l'accès basé sur des identifications, vous devez fournir les informations d'identifications dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/`*`key-name`* , `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`. Pour plus d'informations sur l'étiquetage des ressources Amazon EKS, consultez the section called "Balisage de vos ressources ". Pour plus d'informations sur les actions dans lesquelles vous pouvez utiliser des balises avec des clés de condition, consultez Actions définies par Amazon EKS dans Référence de l'autorisation de service.

Rôles IAM Amazon EKS

Un rôle IAM est une entité de votre AWS compte qui possède des autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec Amazon EKS

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à l'aide de la fédération, endosser un rôle IAM ou encore pour endosser un rôle intercompte. Vous obtenez des

informations d'identification de sécurité temporaires en appelant AWS des opérations d'API STS telles que AssumeRoleou GetFederationToken.

Amazon EKS est compatible avec l'utilisation des informations d'identification temporaires.

Rôles liés à un service

Les rôles liés aux AWS services permettent aux services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre compte IAM et sont la propriété du service. Un administrateur peut consulter mais ne peut pas modifier les autorisations pour les rôles liés à un service.

Amazon EKS prend en charge les rôles liés à un service. Pour plus d'informations sur la création ou la gestion des rôles liés à un service Amazon EKS, consultez the section called "Rôles liés à un service".

Rôles de service

Cette fonction permet à un service d'endosser une fonction du service en votre nom. Ce rôle autorise le service à accéder à des ressources d'autres services pour effectuer une action en votre nom. Les rôles de service s'affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu'un administrateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

Amazon EKS prend en charge les rôles de service. Pour plus d'informations, consultez the section called "Rôle IAM du cluster" et the section called "Rôle IAM de nœud".

Choix d'un rôle IAM dans Amazon EKS

Lorsque vous créez une ressource de cluster dans Amazon EKS, vous devez choisir un rôle pour permettre à Amazon EKS d'accéder à plusieurs autres AWS ressources en votre nom. Si vous avez déjà créé un rôle de service, Amazon EKS vous propose une liste de rôles parmi lesquels choisir. Il est important de choisir un rôle auquel les politiques gérées par Amazon EKS sont associées. Pour plus d'informations, consultez the section called "Recherche d'un rôle de cluster existant" et the section called "Recherche d'un rôle de nœud existant".

Exemples de politiques basées sur l'identité d'Amazon EKS

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou à modifier des ressources Amazon EKS. Ils ne peuvent pas non plus effectuer de tâches à l'aide de la AWS Management

Console AWS CLI ou de AWS l'API. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, veuillez consulter Création de politiques dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Lorsque vous créez un cluster Amazon EKS, le principal IAM qui crée le cluster reçoit automatiquement des `system:masters` autorisations dans la configuration du contrôle d'accès basé sur les rôles (RBAC) du cluster dans le plan de contrôle Amazon EKS. Ce principal n'apparaît dans aucune configuration visible. Assurez-vous donc de savoir quel principal a créé le cluster à l'origine. Pour autoriser des principaux IAM supplémentaires à interagir avec votre cluster, modifiez-le `aws-auth ConfigMap` dans Kubernetes et créez un Kubernetes `rolebinding` ou `clusterrolebinding` avec le nom d'un Kubernetes que vous spécifiez dans le. `group` `aws-auth ConfigMap`

Pour plus d'informations sur l'utilisation du ConfigMap, consultezthe section called “Accès à l'API Kubernetes”.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console Amazon EKS
- Autorisation des utilisateurs IAM à afficher leurs propres autorisations
- Création d'un cluster Kubernetes sur le cloud AWS
- Création d'un cluster Kubernetes local sur un Outpost
- Mise à jour d'un cluster Kubernetes
- Affichage de la liste ou description de tous les clusters

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources Amazon EKS dans votre compte. Ces actions peuvent entraîner des frais pour votre AWS compte. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre AWS compte. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un AWS service spécifique, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root dans AWS votre compte, activez l'authentification MFA pour une sécurité accrue. Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Configuration de l'accès aux API protégé par MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon EKS

Pour accéder à la console Amazon EKS, un principal IAM doit disposer d'un ensemble minimum d'autorisations. Ces autorisations permettent au principal de répertorier et de consulter les informations relatives aux ressources Amazon EKS de votre AWS compte. Si vous créez une politique basée sur l'identité qui est plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les principaux auxquels cette politique est attachée.

Pour garantir que vos principaux IAM puissent toujours utiliser la console Amazon EKS, créez une politique avec votre propre nom unique, par exemple `AmazonEKSAdminPolicy`. Attachez la politique aux principaux. Pour plus d'informations, consultez la rubrique Ajout et suppression d'autorisations basées sur l'identité IAM du Guide de l'utilisateur IAM.

> **Important**
>
> L'exemple de politique suivant permet à un principal d'afficher des informations dans l'onglet Configuration sur la console. Pour afficher les informations dans les onglets Vue d'ensemble et Ressources du AWS Management Console, le principal doit également disposer des autorisations Kubernetes. Pour de plus amples informations, veuillez consulter the section called “Autorisations requises”.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
```

```
        }
    ]
}
```

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux principaux qui appellent uniquement la AWS CLI ou l' AWS API. Au lieu de cela, autorisez uniquement l'accès aux actions correspondant à l'opération d'API que vous essayez d'effectuer.

Autorisation des utilisateurs IAM à afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de la AWS CLI ou AWS de l'API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws: iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Création d'un cluster Kubernetes sur le cloud AWS

Cet exemple de politique inclut les autorisations minimales requises pour créer un cluster Amazon EKS nommé *my-cluster* dans la *us-west-2* AWS région. Vous pouvez remplacer la AWS région par la AWS région dans laquelle vous souhaitez créer un cluster. Si vous voyez un avertissement indiquant que les actions de votre politique ne prennent pas en charge les autorisations au niveau des ressources et nécessitent que vous choisissiez **`All resources`** de le faire AWS Management Console, il peut être ignoré en toute sécurité. Si votre compte possède déjà le rôle *AWSServiceRoleForAmazonEKS*, vous pouvez supprimer l'action `iam:CreateServiceLinkedRole` de la politique. Si vous avez déjà créé un cluster Amazon EKS dans votre compte, ce rôle existe déjà, sauf si vous l'avez supprimé.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws: iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws: iam::111122223333:role/cluster-role-name"
        }
```

```
    ]
}
```

Création d'un cluster Kubernetes local sur un Outpost

Cet exemple de politique inclut les autorisations minimales requises pour créer un cluster local Amazon EKS nommé *my-cluster* sur un avant-poste de la *us-west-2* AWS région. Vous pouvez remplacer la AWS région par la AWS région dans laquelle vous souhaitez créer un cluster. Si vous voyez un avertissement indiquant que les actions de votre politique ne prennent pas en charge les autorisations au niveau des ressources et nécessitent que vous choisissiez **All resources** de le faire AWS Management Console, il peut être ignoré en toute sécurité. Si votre compte possède déjà le rôle `AWSServiceRoleForAmazonEKSLocalOutpost`, vous pouvez supprimer l'action `iam:CreateServiceLinkedRole` de la politique. Si vous avez déjà créé un cluster local Amazon EKS sur un Outpost de votre compte, ce rôle existe déjà, sauf si vous l'avez supprimé.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws: iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
```

```
                "iam:ListAttachedRolePolicies"
            ]
            "Resource": "arn:aws: iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws: iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        },
    ]
}
```

Mise à jour d'un cluster Kubernetes

Cet exemple de politique inclut l'autorisation minimale requise pour mettre à jour un cluster nommé *my-cluster* dans la région us-west-2. AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws: eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

Affichage de la liste ou description de tous les clusters

Cet exemple de politique inclut les autorisations minimales requises pour répertorier et décrire tous les clusters de votre compte. Un principal IAM doit être capable de répertorier et de décrire les clusters pour utiliser la commande `update-kubeconfig` AWS CLI.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Utilisation des rôles liés à un service pour Amazon EKS

Amazon Elastic Kubernetes Service AWS utilise des rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés au service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Rubriques

- Utilisation de rôles pour les clusters Amazon EKS
- Utilisation de rôles pour les groupes de nœuds Amazon EKS
- Utilisation de rôles pour les profils Amazon EKS Fargate
- Utilisation de rôles pour connecter un cluster Kubernetes à Amazon EKS
- Utilisation de rôles pour les clusters locaux Amazon EKS sur Outpost
- Utilisation des rôles pour le tableau de bord Amazon EKS

Utilisation de rôles pour les clusters Amazon EKS

Amazon Elastic Kubernetes Service AWS utilise des rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés au service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations

définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié au service nommé. `AWSServiceRoleForAmazonEKS` Ce rôle permet à Amazon EKS de gérer les clusters de votre compte. Les politiques jointes permettent au rôle de gérer les ressources suivantes : interfaces réseau, groupes de sécurité, journaux et VPCs.

> **Note**
>
> Le rôle lié à un service `AWSServiceRoleForAmazonEKS` est distinct du rôle requis pour la création de cluster. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM du cluster".

Le rôle lié à un service `AWSServiceRoleForAmazonEKS` approuve les services suivants pour endosser le rôle :

- `eks.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous créez un cluster dans la AWS Management Console AWS CLI ou l' AWS API, Amazon EKS crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un cluster, Amazon EKS crée automatiquement le rôle lié au service à nouveau.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKS`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.

3. Si votre cluster possède des groupes de nœuds ou des profils Fargate, vous devez les supprimer avant de pouvoir supprimer le cluster. Pour plus d'informations, consultez the section called "Suppression" et the section called "Supprimer des profils".
4. Dans la page Clusters, choisissez le cluster à supprimer et cliquez sur Delete (Supprimer).
5. Tapez le nom du cluster dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
6. Répétez cette procédure pour tous les autres clusters de votre compte. Attendez la fin de toutes les opérations de suppression.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonEKS` service. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation de rôles pour les groupes de nœuds Amazon EKS

Amazon EKS utilise des AWS rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés au service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié au service nommé. `AWSServiceRoleForAmazonEKSNodegroup` Ce rôle permet à Amazon EKS de gérer les groupes de nœuds de votre compte. La `AWSServiceRoleForAmazonEKSNodegroup` politique ci-jointe permet au rôle de gérer les ressources suivantes : groupes Auto Scaling, groupes de sécurité, modèles de lancement et profils d'instance IAM. Pour de plus amples informations, veuillez consulter the section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegroup".

Le rôle lié à un service `AWSServiceRoleForAmazonEKSNodegroup` approuve les services suivants pour endosser le rôle :

- `eks-nodegroup.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AWSServiceRoleForAmazonEKSNodegroup

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous CreateNodegroup utilisez la AWS Management Console AWS CLI ou l' AWS API, Amazon EKS crée le rôle lié au service pour vous.

> **Important**
>
> Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle. Si vous utilisiez le service Amazon EKS avant le 1er janvier 2017, date à laquelle il a commencé à prendre

en charge les rôles liés au service, Amazon EKS a créé le AWSService RoleForAmazon EKSNodegroup rôle dans votre compte. Pour en savoir plus, consultez Un nouveau rôle est apparu dans mon compte IAM.

Création d'un rôle lié à un service dans Amazon EKS (API)AWS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous créez un groupe de nœuds gérés dans la AWS Management Console AWS CLI ou l' AWS API, Amazon EKS crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un autre groupe de nœuds gérés, Amazon EKS crée automatiquement le rôle lié au service.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSNodegroup`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sélectionnez l'onglet Compute (Calcul).
4. Dans la section Node Groups (Groupes de nœuds), choisissez le groupe de nœuds à supprimer.
5. Tapez le nom du groupe de nœuds dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
6. Répétez cette procédure pour tous les autres groupes de nœuds du cluster. Attendez la fin de toutes les opérations de suppression.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonEKSNodegroup` service. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation de rôles pour les profils Amazon EKS Fargate

Amazon EKS utilise des AWS rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié au service nommé. `AWSServiceRoleForAmazonEKSForFargate` Ce rôle permet à Amazon EKS Fargate de configurer le réseau VPC requis pour les Fargate Pods. Les politiques attachées permettent au rôle de créer et de supprimer des interfaces réseau Elastic et de décrire les interfaces et les ressources réseau Elastic.

Le rôle lié à un service `AWSServiceRoleForAmazonEKSForFargate` approuve les services suivants pour endosser le rôle :

- `eks-fargate.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSForFargateServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous créez un profil Fargate dans AWS Management Console la AWS CLI ou l' AWS API, Amazon EKS crée le rôle lié au service pour vous.

> **Important**
>
> Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle. Si vous utilisiez le service Amazon EKS avant le 13 décembre 2019, date à laquelle il a commencé à prendre en charge les rôles liés au service, Amazon EKS a créé le rôle Fargate dans votre AWSService

RoleForAmazon EKSFor compte. Pour plus d'informations, consultez Un nouveau rôle est apparu dans mon compte IAM.

Création d'un rôle lié à un service dans Amazon EKS (API)AWS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous créez un profil Fargate dans AWS Management Console la AWS CLI ou l' AWS API, Amazon EKS crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un autre groupe de nœuds gérés, Amazon EKS crée automatiquement le rôle lié au service.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSForFargate`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.

2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sur la page Clusters, sélectionnez votre cluster.
4. Sélectionnez l'onglet Compute (Calcul).
5. S'il existe des profils Fargate dans la section Profils Fargate, sélectionnez chacun individuellement, puis choisissez Delete (Supprimer).
6. Tapez le nom du profile dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
7. Répétez cette procédure pour tous les autres profils Fargate dans le cluster et pour tous les autres clusters de votre compte.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au service AWSService RoleForAmazon EKSFor Fargate. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation de rôles pour connecter un cluster Kubernetes à Amazon EKS

Amazon EKS utilise des AWS rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié au service nommé. `AWSServiceRoleForAmazonEKSConnector` Ce rôle permet à Amazon EKS de connecter des clusters Kubernetes. Les politiques jointes permettent au rôle de gérer les ressources nécessaires pour se connecter à votre cluster Kubernetes enregistré.

Le rôle lié à un service `AWSServiceRoleForAmazonEKSConnector` approuve les services suivants pour endosser le rôle :

- `eks-connector.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSConnectorServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service pour connecter un cluster. Lorsque vous connectez un cluster dans la AWS Management Console AWS CLI ou l' AWS API`eksctl`, Amazon EKS crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous connectez un cluster, Amazon EKS crée à nouveau le rôle lié à un service pour vous.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSConnector`. Une fois que vous avez créé un rôle lié à un service,

vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sur la page Clusters, sélectionnez votre cluster.
4. Sélectionnez les onglets Annuler l'inscription puis Ok.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au AWSService RoleForAmazon EKSConnector service. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Utilisation de rôles pour les clusters locaux Amazon EKS sur Outpost

Amazon Elastic Kubernetes Service AWS utilise des rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon

EKS. Les rôles liés au service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié au service nommé. `AWSServiceRoleForAmazonEKSLocalOutpost` Ce rôle permet à Amazon EKS de gérer les clusters locaux de votre compte. Les politiques jointes permettent au rôle de gérer les ressources suivantes : interfaces réseau, groupes de sécurité, journaux et EC2 instances Amazon.

> **Note**
>
> Le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost` est distinct du rôle requis pour la création de cluster. Pour de plus amples informations, veuillez consulter the section called “Rôle IAM du cluster”.

Le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost` approuve les services suivants pour endosser le rôle :

- `outposts.eks-local.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous créez un cluster dans la AWS Management Console AWS CLI ou l' AWS API, Amazon EKS crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un cluster, Amazon EKS crée automatiquement le rôle lié au service à nouveau.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas modifier le nom du rôle car différentes entités peuvent y faire référence. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Sélectionnez Amazon EKS Clusters (Clusters Amazon EKS) dans le panneau de navigation de gauche.
3. Si votre cluster possède des groupes de nœuds ou des profils Fargate, vous devez les supprimer avant de pouvoir supprimer le cluster. Pour plus d'informations, consultez the section called "Suppression" et the section called "Supprimer des profils".
4. Dans la page Clusters, choisissez le cluster à supprimer et cliquez sur Delete (Supprimer).
5. Tapez le nom du cluster dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
6. Répétez cette procédure pour tous les autres clusters de votre compte. Attendez la fin de toutes les opérations de suppression.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonEKSLocalOutpost` service. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation des rôles pour le tableau de bord Amazon EKS

Le tableau de bord Amazon EKS utilise ce rôle lié au service pour agréger les informations sur les ressources du cluster provenant de plusieurs régions et comptes. Le tableau de bord s'intègre à AWS Organizations pour lire en toute sécurité les informations relatives aux multiples comptes de votre organisation.

Pour en savoir plus sur le tableau de bord Amazon EKS, consultezthe section called "Tableau de bord Amazon EKS".

Contexte

Amazon EKS utilise des AWS rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un

service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon EKS, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié au service nommé. `AWSServiceRoleForAmazonEKSDashboard` Ce rôle permet à Amazon EKS de générer et d'afficher le tableau de bord EKS, y compris des informations agrégées sur les ressources du cluster. La `AmazonEKSDashboardServiceRolePolicy` politique ci-jointe permet au rôle de gérer les ressources suivantes : groupes Auto Scaling, groupes de sécurité, modèles de lancement et profils d'instance IAM. Pour de plus amples informations, veuillez consulter the section called “AWS politique gérée : Amazon EKSDashboard ServiceRolePolicy”.

Le rôle lié à un service `AWSServiceRoleForAmazonEKSDashboard` approuve les services suivants pour endosser le rôle :

- `dashboard.eks.amazonaws.com`

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSDashboard ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service. Lorsque vous activez le tableau de bord dans la AWS console, Amazon EKS crée pour vous le rôle lié au service.

Pour plus d'informations sur l'activation du tableau de bord EKS, consultezthe section called "Configuration des organisations".

> **Important**
>
> Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSDashboard`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour plus d'informations sur la désactivation du tableau de bord EKS, consultezthe section called "Configuration des organisations".

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonEKSDashboard` service. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Rôle IAM d'exécution d'Amazon EKS Pod

Le rôle d'exécution Amazon EKS Pod est requis pour exécuter des pods sur l'infrastructure AWS Fargate.

Lorsque votre cluster crée des pods sur l'infrastructure AWS Fargate, les composants exécutés sur l'infrastructure Fargate doivent effectuer des appels en votre nom. AWS APIs Cela leur permet d'effectuer des actions telles que l'extraction d'images de conteneurs depuis Amazon ECR ou le routage des journaux vers d'autres AWS services. Le rôle d'exécution Amazon EKS Pod fournit les autorisations IAM nécessaires à cette fin.

Lorsque vous créez un profil Fargate, vous devez spécifier un rôle d'exécution Pod pour les composants Amazon EKS qui s'exécutent sur l'infrastructure Fargate à l'aide du profil. Ce rôle est ajouté au contrôle d'accès basé sur les rôles Kubernetes (RBAC) du cluster à des fins d'autorisation. Cela permet au système `kubelet` qui s'exécute sur l'infrastructure Fargate de s'enregistrer auprès de votre cluster Amazon EKS afin qu'il puisse apparaître dans votre cluster en tant que nœud.

> **Note**
>
> Le profil Fargate doit avoir un rôle IAM différent de celui EC2 des groupes de nœuds Amazon.

> **Important**
>
> Les conteneurs exécutés dans le Fargate Pod ne peuvent pas assumer les autorisations IAM associées à un rôle d'exécution du Pod. Pour autoriser les conteneurs de votre Fargate Pod à accéder à d' AWS autres services, vous devez utiliser des rôles IAM pour les comptes de service.

Avant de créer un profil Fargate, vous devez créer un rôle IAM auprès d'Amazon. EKSFargate PodExecutionRolePolicy

Vérifiez si un rôle d'exécution de Pod existant est correctement configuré

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà un rôle d'exécution Amazon EKS Pod correctement configuré. Pour éviter un problème de sécurité adjoint confus, il est important que le rôle limite l'accès en fonction `SourceArn` de. Vous pouvez modifier le rôle d'exécution si nécessaire pour inclure la prise en charge des profils Fargate sur d'autres clusters.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, recherchez la liste des rôles pour Amazon EKSFargate PodExecutionRole. Si le rôle n'existe pas, reportez-vous the section called "Création du rôle d'exécution Amazon EKS Pod" à la section pour le créer. Si le rôle existe, sélectionnez-le.
4. Sur la EKSFargate PodExecutionRole page Amazon, procédez comme suit :
   a. Choisissez Autorisations.
   b. Assurez-vous que la politique gérée par EKSFargatePodExecutionRolePolicyAmazon est attachée au rôle.
   c. Choisissez Trust Relationships (Relations d'approbation).
   d. Choisissez Edit trust policy (Modifier la politique).
5. Dans la page Edit trust policy (Modifier la politique), vérifiez que la relation d'approbation contient la politique suivante, ainsi qu'une ligne pour les profils Fargate sur votre cluster. Dans ce cas, sélectionnez Cancel (Annuler).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/
my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Si la politique correspond mais qu'aucune ligne ne spécifie les profils Fargate de votre cluster, vous pouvez ajouter la ligne suivante en haut de l'objet. `ArnLike` Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster, *`111122223333`* par votre identifiant de compte et *`my-cluster`* par le nom de votre cluster.

```
"aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

Si la politique ne correspond pas, copiez la politique précédente complète dans le formulaire et choisissez Mettre à jour la politique. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Si vous souhaitez utiliser le même rôle dans toutes les AWS régions de votre compte, remplacez *`region-code`* par`*`. Remplacez *`111122223333`* par l'ID de votre compte et *`my-cluster`* par le nom de votre cluster. Si vous souhaitez utiliser le même rôle pour tous les clusters de votre compte, remplacez *`my-cluster`* par `*`.

Création du rôle d'exécution Amazon EKS Pod

Si vous ne possédez pas encore le rôle d'exécution Amazon EKS Pod pour votre cluster, vous pouvez utiliser la AWS Management Console ou la AWS CLI pour le créer.

AWS Management Console

a. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
b. Dans le volet de navigation de gauche, choisissez Rôles.
c. Sur la page Rôles, choisissez Créer un rôle.

d. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :

   i. Dans la section Type d'entité fiable, sélectionnez AWS service.

   ii. Dans la liste déroulante Cas d'utilisation d'autres AWS services, sélectionnez EKS.

   iii. Choisissez EKS - Fargate Pod.

   iv. Choisissez Suivant.

e. Sur la page Add permissions (Ajouter des autorisations), sélectionnez Next (Suivant).

f. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   i. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSFargatePodExecutionRole`.

   ii. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

   iii. Sélectionnez Créer un rôle.

g. Sur la page Rôles, recherchez la liste des rôles pour Amazon EKSFargate PodExecutionRole. Choisissez le rôle.

h. Sur la EKSFargate PodExecutionRole page Amazon, procédez comme suit :

   i. Choisissez Trust Relationships (Relations d'approbation).

   ii. Choisissez Edit trust policy (Modifier la politique).

i. Dans la page Edit trust policy (Modifier la politique), procédez comme suit :

   i. Copiez le contenu suivant et collez-le dans le formulaire Edit trust policy (Modifier la politique). Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Si vous souhaitez utiliser le même rôle dans toutes les AWS régions de votre compte, remplacez *`region-code`* par`*`. Remplacez *`111122223333`* par l'ID de votre compte et *`my-cluster`* par le nom de votre cluster. Si vous souhaitez utiliser le même rôle pour tous les clusters de votre compte, remplacez *`my-cluster`* par `*`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
```

```
            "aws:SourceArn": "arn:aws: eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ii. Choisissez Mettre à jour une politique.

AWS CLI

a. Copiez le contenu suivant et collez-le dans un fichier nommé `pod-execution-role-trust-policy.json`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Si vous souhaitez utiliser le même rôle dans toutes les AWS régions de votre compte, remplacez *`region-code`* par`*`. Remplacez *`111122223333`* par l'ID de votre compte et *`my-cluster`* par le nom de votre cluster. Si vous souhaitez utiliser le même rôle pour tous les clusters de votre compte, remplacez *`my-cluster`* par `*`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws: eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez un rôle IAM d'exécution du Pod.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. Attachez la politique IAM gérée par Amazon EKS au rôle.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Rôle IAM du connecteur Amazon EKS

Vous pouvez connecter des clusters Kubernetes pour les afficher dans votre. AWS Management Console Pour vous connecter à un cluster Kubernetes, créez un rôle IAM.

Rechercher un rôle de connecteur EKS existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de connecteur Amazon EKS.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `AmazonEKSConnectorAgentRole`. Si un rôle incluant `AmazonEKSConnectorAgentRole` n'existe pas, reportez-vous the section called "Création du rôle de l'agent Amazon EKS Connector" à la section Création du rôle. Si un rôle qui inclut `AmazonEKSConnectorAgentRole` existe, sélectionnez le rôle pour afficher les politiques attachées.
4. Choisissez Permissions (Autorisations).
5. Assurez-vous que la politique EKSConnector AgentPolicy gérée par Amazon est attachée au rôle. Si la politique est attachée, votre rôle de connecteur Amazon EKS est configuré correctement.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique de confiance et choisissez Mettre à jour la politique.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Création du rôle de l'agent Amazon EKS Connector

Vous pouvez utiliser le AWS Management Console ou AWS CloudFormation pour créer le rôle d'agent du connecteur.

AWS CLI

a. Créez le fichier nommé `eks-connector-agent-trust-policy.json` qui contient le JSON suivant à utiliser pour le rôle IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Créez le fichier nommé `eks-connector-agent-policy.json` qui contient le JSON suivant à utiliser pour le rôle IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws: eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

c. Créez le rôle d'agent Amazon EKS Connector à l'aide de la politique d'approbation et de la politique que vous avez créées dans les listes précédentes.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

d. Attachez la politique à votre rôle d'agent Amazon EKS Connector.

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

a. Enregistrez le AWS CloudFormation modèle suivant dans un fichier texte sur votre système local.

> **Note**
>
> Ce modèle crée également le rôle lié à un service qui serait autrement créé lorsque l'API `registerCluster` est appelée. Consultez the section called "Rôle de connecteur de cluster" pour plus de détails.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
```

```
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS
 services.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

b. Ouvrez la AWS CloudFormation console.
c. Choisissez Créer une pile avec de nouvelles ressources (standard).
d. Pour Specify template (Spécifier un modèle), sélectionnez Upload a template file (Télécharger un fichier de modèle), puis choisissez Choose file (Choisir un fichier).
e. Choisissez le fichier que vous avez créé précédemment, puis choisissez Next (Suivant).
f. Pour Stack name (Nom de la pile), saisissez un nom pour votre rôle, par exemple `eksConnectorAgentRole`, puis choisissez Next (Suivant).
g. Sur la page Configurer les options de pile, choisissez Next (Suivant).
h. Sur la page Vérification, vérifiez vos informations, reconnaissez que la pile peut créer des ressources IAM, puis choisissez Create stack (Créer la pile).

AWS politiques gérées pour Amazon Elastic Kubernetes Service

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte

toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'un nouveau AWS service est lancé ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : Amazoneks_CNI_Policy

Vous pouvez attacher `AmazonEKS_CNI_Policy` à vos entités IAM. Avant de créer un groupe de EC2 nœuds Amazon, cette politique doit être attachée au rôle IAM du nœud ou à un rôle IAM spécifiquement utilisé par le plugin Amazon VPC CNI pour Kubernetes. Ceci lui permet d'effectuer des actions en votre nom. Nous vous recommandons d'associer la politique à un rôle utilisé uniquement par le plugin. Pour plus d'informations, consultez the section called "CNI Amazon VPC" et the section called "Configuration pour IRSA".

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2:*NetworkInterface`**et **`ec2:*PrivateIpAddresses`** — Permet au plugin Amazon VPC CNI d'effectuer des actions telles que le provisionnement d'interfaces réseau élastiques et d'adresses IP pour les pods afin de fournir un réseau aux applications exécutées dans Amazon EKS.
- **`ec2`**actions de lecture — Permet au plugin Amazon VPC CNI d'effectuer des actions telles que décrire des instances et des sous-réseaux pour voir le nombre d'adresses IP gratuites dans vos sous-réseaux Amazon VPC. Le VPC CNI peut utiliser les adresses IP libres de chaque sous-réseau pour sélectionner les sous-réseaux avec le plus grand nombre d'adresses IP libres à utiliser lors de la création d'une interface elastic network.

Pour consulter la dernière version du document de politique JSON, consultez Amazoneks_CNI_Policy dans le Managed Policy Reference Guide. AWS

AWS politique gérée : Amazon EKSCluster Policy

Vous pouvez attacher `AmazonEKSClusterPolicy` à vos entités IAM. Avant de créer un cluster, vous devez disposer d'un rôle IAM de cluster avec la politique ci-jointe. Les clusters Kubernetes

gérés par Amazon EKS appellent d'autres AWS services en votre nom. Ils le font pour gérer les ressources que vous utilisez avec le service.

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`autoscaling`**— Lisez et mettez à jour la configuration d'un groupe Auto Scaling. Ces autorisations ne sont pas utilisées par Amazon EKS mais restent incluses dans la politique de rétrocompatibilité.
- **`ec2`**— Travaillez avec des volumes et des ressources réseau associés aux EC2 nœuds Amazon. Ceci est nécessaire pour que le plan de contrôle Kubernetes puisse joindre des instances à un cluster et approvisionner et gérer dynamiquement les volumes Amazon EBS demandés par des volumes persistants Kubernetes.
- **`ec2`**- Supprimez les interfaces réseau élastiques créées par le VPC CNI. Cela est nécessaire pour qu'EKS puisse nettoyer les interfaces réseau élastiques qui restent en place en cas de fermeture inattendue du VPC CNI.
- **`elasticloadbalancing`**— Travaillez avec des équilibreurs de charge élastiques et ajoutez-y des nœuds comme cibles. Ceci est nécessaire pour que le plan de contrôle Kubernetes puisse allouer de manière dynamique les Elastic Load Balancer demandés par les services Kubernetes.
- **`iam`**— Créez un rôle lié à un service. Ceci est nécessaire pour que le plan de contrôle Kubernetes puisse allouer de manière dynamique les Elastic Load Balancer demandés par les services Kubernetes.
- **`kms`**— Lit une clé depuis AWS KMS. Cette condition est nécessaire pour que le plan de contrôle de Kubernetes prenne en charge le chiffrement des secrets Kubernetes stockés dans `etcd`.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSCluster Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSDashboard ConsoleReadOnly

Vous pouvez attacher `AmazonEKSDashboardConsoleReadOnly` à vos entités IAM.

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`eks`**- Accès en lecture seule aux données du tableau de bord EKS, aux ressources et aux informations sur les versions du cluster. Cela permet de consulter les métriques liées à EKS et les détails de configuration du cluster.
- **`organizations`**- Accès en lecture seule aux informations des AWS Organizations, notamment :
  - Afficher les détails de l'organisation et l'accès aux services
  - Répertorier les racines, les comptes et les unités organisationnelles
  - Afficher la structure de l'organisation

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSDashboard ConsoleReadOnly dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSFargate PodExecutionRolePolicy

Vous pouvez attacher `AmazonEKSFargatePodExecutionRolePolicy` à vos entités IAM. Avant de créer un profil Fargate, vous devez créer un rôle d'exécution Fargate Pod et y associer cette politique. Pour plus d'informations, consultez the section called "Étape 2 : Création d'un rôle d'exécution Fargate Pod" et the section called "Définir des profils".

Cette politique accorde au rôle les autorisations lui permettant d'accéder aux autres ressources de AWS service requises pour exécuter les pods Amazon EKS sur Fargate.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ecr`**— Permet aux pods exécutés sur Fargate d'extraire des images de conteneurs stockées dans Amazon ECR.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSFargate PodExecutionRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSFor FargateServiceRolePolicy

Vous ne pouvez pas vous associer `AmazonEKSForFargateServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter `AWSServiceRoleforAmazonEKSForFargate`.

Cette politique accorde les autorisations nécessaires à Amazon EKS pour exécuter des tâches Fargate. La politique n'est utilisée que si vous avez des nœuds Fargate.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`ec2`**— Créez et supprimez des interfaces réseau élastiques et décrivez les interfaces réseau élastiques et leurs ressources. Cela est nécessaire pour que le service Amazon EKS Fargate puisse configurer le réseau VPC requis pour les Fargate Pods.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSFor FargateServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSCompute Policy

Vous pouvez attacher `AmazonEKSComputePolicy` à vos entités IAM. Vous pouvez associer cette politique au rôle IAM de votre cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique accorde les autorisations requises à Amazon EKS pour créer et gérer des EC2 instances pour le cluster EKS, ainsi que les autorisations IAM nécessaires à la configuration EC2. Cette politique autorise également Amazon EKS à créer le rôle lié au service EC2 Spot en votre nom.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`**Autorisations :
  - `ec2:CreateFleet`et `ec2:RunInstances` - Permet de créer des EC2 instances et d'utiliser des EC2 ressources spécifiques (images, groupes de sécurité, sous-réseaux) pour les nœuds du cluster EKS.
  - `ec2:CreateLaunchTemplate`- Permet de créer des modèles de EC2 lancement pour les nœuds du cluster EKS.
  - La politique inclut également des conditions visant à restreindre l'utilisation de ces EC2 autorisations aux ressources étiquetées avec le nom du cluster EKS et d'autres balises pertinentes.

- `ec2:CreateTags`- Permet d'ajouter des balises aux EC2 ressources créées par les `CreateLaunchTemplate` actions `CreateFleetRunInstances`, et.
- **`iam`**Autorisations :
  - `iam:AddRoleToInstanceProfile`- Permet d'ajouter un rôle IAM au profil d'instance de calcul EKS.
  - `iam:PassRole`- Permet de transmettre les rôles IAM nécessaires au EC2 service.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSCompute Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSNetworking Policy

Vous pouvez attacher `AmazonEKSNetworkingPolicy` à vos entités IAM. Vous pouvez associer cette politique au rôle IAM de votre cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique est conçue pour accorder les autorisations nécessaires à Amazon EKS pour créer et gérer des interfaces réseau pour le cluster EKS, permettant au plan de contrôle et aux nœuds de travail de communiquer et de fonctionner correctement.

Détails de l'autorisation

Cette politique accorde les autorisations suivantes pour permettre à Amazon EKS de gérer les interfaces réseau du cluster :

- **`ec2`**Autorisations d'interface réseau :
  - `ec2:CreateNetworkInterface`- Permet de créer des interfaces EC2 réseau.
  - La politique inclut des conditions visant à restreindre l'utilisation de cette autorisation aux interfaces réseau étiquetées avec le nom du cluster EKS et le nom du nœud Kubernetes CNI.
  - `ec2:CreateTags`- Permet d'ajouter des balises aux interfaces réseau créées par l'`CreateNetworkInterface`action.
- **`ec2`**Autorisations de gestion de l'interface réseau :
  - `ec2:AttachNetworkInterface`, `ec2:DetachNetworkInterface` - Permet d'attacher et de détacher des interfaces réseau aux EC2 instances.
  - `ec2:UnassignPrivateIpAddresses`,`ec2:UnassignIpv6Addresses`,`ec2:AssignPrivateIpAd` `ec2:AssignIpv6Addresses` - Permet de gérer les assignations d'adresses IP des interfaces réseau.

- Ces autorisations sont limitées aux interfaces réseau étiquetées avec le nom du cluster EKS.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSNetworking Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSBlock StoragePolicy

Vous pouvez attacher `AmazonEKSBlockStoragePolicy` à vos entités IAM. Vous pouvez associer cette politique au rôle IAM de votre cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique accorde les autorisations nécessaires à Amazon EKS pour créer, gérer et gérer des EC2 volumes et des instantanés pour le cluster EKS, permettant ainsi au plan de contrôle et aux nœuds de travail de provisionner et d'utiliser un stockage persistant conformément aux charges de travail Kubernetes.

Détails de l'autorisation

Cette politique IAM accorde les autorisations suivantes pour permettre à Amazon EKS de gérer les EC2 volumes et les instantanés :

- **`ec2`**Autorisations de gestion des volumes :
  - `ec2:AttachVolume`,, `ec2:DetachVolumeec2:ModifyVolume`, `ec2:EnableFastSnapshotRestores` - Permet de joindre, de détacher, de modifier et d'activer des restaurations instantanées rapides pour les EC2 volumes.
  - Ces autorisations sont limitées aux volumes marqués avec le nom du cluster EKS.
  - `ec2:CreateTags`- Permet d'ajouter des balises aux EC2 volumes et aux instantanés créés par les `CreateSnapshot` actions `CreateVolume` et.
- **`ec2`**Autorisations de création de volumes :
  - `ec2:CreateVolume`- Permet de créer de nouveaux EC2 volumes.
  - La politique inclut des conditions visant à restreindre l'utilisation de cette autorisation aux volumes marqués avec le nom du cluster EKS et d'autres balises pertinentes.
  - `ec2:CreateSnapshot`- Permet de créer de nouveaux instantanés de EC2 volumes.
  - La politique inclut des conditions visant à restreindre l'utilisation de cette autorisation aux instantanés marqués avec le nom du cluster EKS et d'autres balises pertinentes.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSBlock StoragePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSLoad BalancingPolicy

Vous pouvez attacher `AmazonEKSLoadBalancingPolicy` à vos entités IAM. Vous pouvez associer cette politique au rôle IAM de votre cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique IAM accorde les autorisations nécessaires à Amazon EKS pour travailler avec différents AWS services afin de gérer les Elastic Load Balancers (ELBs) et les ressources associées.

Détails de l'autorisation

Les principales autorisations accordées par cette politique sont les suivantes :

- **`elasticloadbalancing`**: Permet de créer, de modifier et de gérer des équilibreurs de charge élastiques et des groupes cibles. Cela inclut les autorisations permettant de créer, de mettre à jour et de supprimer des équilibreurs de charge, des groupes cibles, des écouteurs et des règles.
- **`ec2`**: Permet de créer et de gérer des groupes de sécurité, nécessaires au plan de contrôle Kubernetes pour joindre des instances à un cluster et gérer les volumes Amazon EBS. Permet également de décrire et de répertorier EC2 des ressources telles que des instances VPCs, des sous-réseaux, des groupes de sécurité et d'autres ressources réseau.
- **`iam`**: Permet de créer un rôle lié à un service pour Elastic Load Balancing, qui est nécessaire au provisionnement dynamique du plan de contrôle Kubernetes. ELBs
- **`kms`**: Permet de lire une clé depuis AWS KMS, qui est requise pour que le plan de contrôle Kubernetes prenne en charge le chiffrement des secrets Kubernetes stockés dans etcd.
- **`wafv2`**et **`shield`**: Permet d'associer et de dissocier le Web ACLs et de créer/supprimer des protections AWS Shield pour les Elastic Load Balancers.
- **`cognito-idp`**, **`acm`**, et **`elasticloadbalancing`**: accorde des autorisations pour décrire les clients du pool d'utilisateurs, répertorier et décrire les certificats, et décrire les groupes cibles, qui sont nécessaires au plan de contrôle Kubernetes pour gérer les Elastic Load Balancers.

La politique inclut également plusieurs vérifications de condition pour garantir que les autorisations sont limitées au cluster EKS spécifique géré, à l'aide de la `eks:eks-cluster-name` balise.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSLoad BalancingPolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSService Policy

Vous pouvez attacher `AmazonEKSServicePolicy` à vos entités IAM. Les clusters créés avant le 16 avril 2020 nécessitaient la création d'un rôle IAM et l'ajout de cette politique. Les clusters créés le 16 avril 2020 ou après cette date ne vous obligent pas à créer un rôle ni à attribuer cette politique. Lorsque vous créez un cluster à l'aide d'un principal IAM `iam:CreateServiceLinkedRole` autorisé, le rôle lié au service AWSServiceRoleforAmazonEKS est automatiquement créé pour vous. Le rôle lié à un service est EKSService RolePolicy associé à la politique gérée suivante : Amazon.

Cette politique permet à Amazon EKS de créer et de gérer les ressources nécessaires à l'exploitation des clusters Amazon EKS.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`eks`**— Mettez à jour la version Kubernetes de votre cluster après avoir lancé une mise à jour. Cette autorisation n'est pas utilisée par Amazon EKS mais reste inscrite dans la politique de rétrocompatibilité.
- **`ec2`**— Travaillez avec les interfaces réseau élastiques et d'autres ressources et balises réseau. Cela est requis par Amazon EKS pour configurer la mise en réseau qui facilite la communication entre les nœuds et le plan de contrôle Kubernetes. Lisez les informations sur les groupes de sécurité. Mettez à jour les balises des groupes de sécurité.
- **`route53`**— Associez un VPC à une zone hébergée. Cela est requis par Amazon EKS pour activer la mise en réseau de points de terminaison privés pour votre serveur d'API de cluster Kubernetes.
- **`logs`**— Enregistrez les événements. Cela est nécessaire pour qu'Amazon EKS puisse envoyer les journaux du plan de contrôle Kubernetes à. CloudWatch
- **`iam`**— Créez un rôle lié à un service. Ceci est nécessaire pour qu'Amazon EKS puisse créer le rôle lié à un service the section called "Autorisations du rôle lié à un service pour Amazon EKS" en votre nom.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSService Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSService RolePolicy

Vous ne pouvez pas vous associer `AmazonEKSServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Autorisations du rôle lié à un service pour Amazon EKS". Lorsque vous créez un cluster à l'aide d'un principal IAM `iam:CreateServiceLinkedRole` autorisé, le rôle lié au service AWSServiceRoleforAmazonEKS est automatiquement créé pour vous et cette politique y est associée.

Cette politique permet au rôle lié au service d'appeler les AWS services en votre nom.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`ec2`**— Créez et décrivez les interfaces réseau Elastic et EC2 les instances Amazon, le groupe de sécurité du cluster et le VPC nécessaires à la création d'un cluster. Pour de plus amples informations, veuillez consulter the section called "Exigences de groupe de sécurité". Lisez les informations sur les groupes de sécurité. Mettez à jour les balises des groupes de sécurité. Lisez les informations sur les réservations de capacité à la demande.
- **`ec2`**Mode automatique : mettez fin aux EC2 instances créées par le mode automatique EKS. Pour de plus amples informations, veuillez consulter Mode automatique EKS.
- **`iam`**— Répertorie toutes les politiques gérées associées à un rôle IAM. Cette condition est nécessaire pour permettre à Amazon EKS de répertorier et valider toutes les politiques et autorisations gérées requises pour créer un cluster.
- Associer un VPC à une zone hébergée : cela est requis par Amazon EKS pour activer la mise en réseau de points de terminaison privés pour votre serveur d'API de cluster Kubernetes.
- Journaliser les événements : cela est nécessaire pour qu'Amazon EKS puisse envoyer les journaux du plan de contrôle Kubernetes à. CloudWatch
- Put metric : cela est nécessaire pour qu'Amazon EKS puisse envoyer les journaux du plan de contrôle Kubernetes à. CloudWatch
- **`eks`**- Gérez les entrées et les politiques d'accès au cluster, ce qui permet de contrôler avec précision les personnes autorisées à accéder aux ressources EKS et les actions qu'elles peuvent effectuer. Cela inclut l'association de politiques d'accès standard pour les opérations de calcul, de mise en réseau, d'équilibrage de charge et de stockage.

- **`elasticloadbalancing`**- Créez, gérez et supprimez les équilibreurs de charge et leurs composants (écouteurs, groupes cibles, certificats) associés aux clusters EKS. Consultez les attributs et l'état de santé de l'équilibreur de charge.
- **`events`**- Créez et gérez des EventBridge règles pour la surveillance EC2 et les événements de AWS santé liés aux clusters EKS, permettant des réponses automatisées aux modifications de l'infrastructure et aux alertes de santé.
- **`iam`**- Gérez les profils d' EC2 instance avec le préfixe « eks », y compris la création, la suppression et l'association de rôles, nécessaires à la gestion des nœuds EKS.
- **`pricing`**& **`shield`**- Accédez aux informations AWS tarifaires et à l'état de protection du Shield, permettant de gérer les coûts et de bénéficier de fonctionnalités de sécurité avancées pour les ressources EKS.
- Nettoyage des ressources : supprimez en toute sécurité les ressources étiquetées EKS, notamment les volumes, les instantanés, les modèles de lancement et les interfaces réseau lors des opérations de nettoyage du cluster.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSService RolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSVPCResource Controller

Vous pouvez associer la politique `AmazonEKSVPCResourceController` à vos identités IAM. Si vous utilisez des groupes de sécurité pour les pods, vous devez associer cette politique au rôle IAM de votre cluster Amazon EKS pour effectuer des actions en votre nom.

Cette politique accorde les autorisations de rôle de cluster pour gérer les interfaces réseau Elastic et les adresses IP pour les nœuds.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`**— Gérez les interfaces réseau élastiques et les adresses IP pour prendre en charge les groupes de sécurité Pod et les nœuds Windows.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSVPCResource Controller dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSWorker NodePolicy

Vous pouvez attacher `AmazonEKSWorkerNodePolicy` à vos entités IAM. Vous devez associer cette politique à un rôle IAM de nœud que vous spécifiez lorsque vous créez des EC2 nœuds Amazon qui permettent à Amazon EKS d'effectuer des actions en votre nom. Si vous créez un groupe de nœuds à l'aide de `eksctl`, il crée le rôle IAM de nœud et attache automatiquement cette politique au rôle.

Cette politique accorde aux EC2 nœuds Amazon EKS l'autorisation de se connecter aux clusters Amazon EKS.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`**— Lisez le volume de l'instance et les informations réseau. Cela est nécessaire pour que les nœuds Kubernetes puissent décrire les informations relatives aux EC2 ressources Amazon requises pour que le nœud rejoigne le cluster Amazon EKS.
- **`eks`**— Décrivez éventuellement le cluster dans le cadre de l'amorçage du nœud.
- **`eks-auth:AssumeRoleForPodIdentity`**— Autorise la récupération des informations d'identification pour les charges de travail EKS sur le nœud. Cela est nécessaire pour que l'identité du pod EKS fonctionne correctement.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSWorker NodePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSWorker NodeMinimalPolicy

Vous pouvez associer l'Amazon EKSWorker NodeMinimalPolicy à vos entités IAM. Vous pouvez associer cette politique à un rôle IAM de nœud que vous spécifiez lorsque vous créez des EC2 nœuds Amazon qui permettent à Amazon EKS d'effectuer des actions en votre nom.

Cette politique accorde aux EC2 nœuds Amazon EKS l'autorisation de se connecter aux clusters Amazon EKS. Cette politique comporte moins d'autorisations que celle d'Amazon EKSWorkerNodePolicy.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- `eks-auth:AssumeRoleForPodIdentity`- Autoriser la récupération des informations d'identification pour les charges de travail EKS sur le nœud. Cela est nécessaire pour que l'identité du pod EKS fonctionne correctement.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSWorker NodePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : AWSService RoleForAmazon EKSNodegroup

Vous ne pouvez pas vous associer `AWSServiceRoleForAmazonEKSNodegroup` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Autorisations du rôle lié à un service pour Amazon EKS".

Cette politique accorde au `AWSServiceRoleForAmazonEKSNodegroup` rôle les autorisations qui lui permettent de créer et de gérer des groupes de EC2 nœuds Amazon dans votre compte.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`**— Travaillez avec des groupes de sécurité, des tags, des réservations de capacité et des modèles de lancement. Cela est nécessaire pour les groupes de nœuds gérés par Amazon EKS afin de permettre la configuration de l'accès à distance et de décrire les réservations de capacité qui peuvent être utilisées dans les groupes de nœuds gérés. En outre, les groupes de nœuds gérés par Amazon EKS créent un modèle de lancement en votre nom. Cela permet de configurer le groupe Amazon EC2 Auto Scaling qui soutient chaque groupe de nœuds gérés.
- **`iam`**— Créez un rôle lié à un service et transmettez-le. Ceci est requis par les groupes de nœuds gérés Amazon EKS pour gérer les profils d'instance pour le rôle transmis lors de la création d'un groupe de nœuds gérés. Ce profil d'instance est utilisé par les EC2 instances Amazon lancées dans le cadre d'un groupe de nœuds géré. Amazon EKS doit créer des rôles liés à un service pour d'autres services tels que les groupes Amazon EC2 Auto Scaling. Ces autorisations sont utilisées dans la création d'un groupe de nœuds gérés.
- **`autoscaling`**— Travaillez avec des groupes de sécurité Auto Scaling. Cela est requis par les groupes de nœuds gérés par Amazon EKS pour gérer le groupe Amazon EC2 Auto Scaling qui

soutient chaque groupe de nœuds gérés. Il est également utilisé pour prendre en charge des fonctionnalités telles que l'expulsion des pods lorsque les nœuds sont résiliés ou recyclés lors des mises à jour des groupes de nœuds.

Pour consulter la dernière version du document de politique JSON, consultez AWSServiceRoleForAmazonEKSNodegrouple Guide de référence des politiques AWS gérées.

AWS politique gérée : Amazon EKSDashboard ServiceRolePolicy

Vous ne pouvez pas vous associer `AmazonEKSDashboardServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Autorisations du rôle lié à un service pour Amazon EKS".

Cette politique accorde au `AWSServiceRoleForAmazonEKSDashboard` rôle les autorisations qui lui permettent de créer et de gérer des groupes de EC2 nœuds Amazon dans votre compte.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui autorisent l'accès pour effectuer ces tâches :

- **`organizations`**— Consultez les informations relatives à la structure et aux comptes de vos AWS Organisations. Cela inclut les autorisations permettant de répertorier les comptes de votre organisation, d'afficher les unités organisationnelles et les racines, de répertorier les administrateurs délégués, de consulter les services ayant accès à votre organisation et de récupérer des informations détaillées sur votre organisation et vos comptes.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSDashboard ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EBSCSIDriver Policy

La politique `AmazonEBSCSIDriverPolicy` permet au pilote Amazon EBS Container Storage Interface (CSI) de créer, de modifier, d'attacher, de détacher et de supprimer des volumes en votre nom. Cela inclut la modification des balises sur les volumes existants et l'activation de la restauration rapide des instantanés (FSR) sur les volumes EBS. Il accorde également au pilote EBS CSI les autorisations nécessaires pour créer, restaurer et supprimer des instantanés, ainsi que pour répertorier vos instances, volumes et instantanés.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EBSCSIDriver ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EFSCSIDriver Policy

La politique `AmazonEFSCSIDriverPolicy` permet à l'interface CSI (Amazon EFS Container Storage Interface) de créer et de supprimer des points d'accès en votre nom. Il accorde également au pilote Amazon EFS CSI l'autorisation de répertorier vos points d'accès, vos systèmes de fichiers, vos cibles de montage et vos zones de EC2 disponibilité Amazon.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EFSCSIDriver ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSLocal OutpostClusterPolicy

Vous ne pouvez pas attacher cette politique à des entités IAM. Avant de créer un cluster local, vous devez associer cette politique à votre rôle de cluster. Les clusters Kubernetes gérés par Amazon EKS appellent d'autres AWS services en votre nom. Ils le font pour gérer les ressources que vous utilisez avec le service.

La politique `AmazonEKSLocalOutpostClusterPolicy` inclut les autorisations suivantes :

- **`ec2`**actions de lecture — Permet aux instances du plan de contrôle de décrire les propriétés de la zone de disponibilité, de la table de routage, de l'instance et de l'interface réseau. Autorisations requises pour que les EC2 instances Amazon rejoignent avec succès le cluster en tant qu'instances du plan de contrôle.
- **`ssm`**— Permet à Amazon EC2 Systems Manager de se connecter à l'instance du plan de contrôle, qui est utilisée par Amazon EKS pour communiquer et gérer le cluster local de votre compte.
- **`logs`**— Permet aux instances de transmettre des journaux à Amazon CloudWatch.
- **`secretsmanager`**— Permet aux instances d'obtenir et de supprimer des données de démarrage pour les instances du plan de contrôle en toute sécurité à partir de AWS Secrets Manager.
- **`ecr`**— Permet aux pods et aux conteneurs exécutés sur les instances du plan de contrôle d'extraire des images de conteneurs stockées dans Amazon Elastic Container Registry.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSLocal OutpostClusterPolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSLocal OutpostServiceRolePolicy

Vous ne pouvez pas associer cette politique à vos entités IAM. Lorsque vous créez un cluster à l'aide d'un principal IAM `iam:CreateServiceLinkedRole` autorisé, Amazon EKS crée automatiquement le rôle lié au service AWSServiceRoleforAmazonEKSLocalOutpost pour vous et y associe cette politique. Cette politique permet au rôle lié au service d'appeler des AWS services en votre nom pour les clusters locaux.

La politique `AmazonEKSLocalOutpostServiceRolePolicy` inclut les autorisations suivantes :

- **`ec2`**— Permet à Amazon EKS de travailler avec des ressources de sécurité, de réseau et autres pour lancer et gérer avec succès les instances du plan de contrôle dans votre compte.
- **`ssm`**, **`ssmmessages`** — Permet à Amazon EC2 Systems Manager de se connecter aux instances du plan de contrôle, qui sont utilisées par Amazon EKS pour communiquer et gérer le cluster local de votre compte.
- **`iam`**— Permet à Amazon EKS de gérer le profil d'instance associé aux instances du plan de contrôle.
- **`secretsmanager`**- Permet à Amazon EKS de placer les données de démarrage des instances du plan de contrôle dans AWS Secrets Manager afin qu'elles puissent être référencées de manière sécurisée lors du démarrage des instances.
- **`outposts`**— Permet à Amazon EKS d'obtenir les informations d'Outpost depuis votre compte afin de lancer avec succès un cluster local dans un Outpost.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSLocal OutpostServiceRolePolicy dans le AWS Managed Policy Reference Guide.

Amazon EKS met à jour les politiques AWS gérées

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour Amazon EKS depuis que ce service a commencé à suivre ces modifications.

Pour recevoir des notifications concernant toutes les modifications apportées au fichier source sur cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante avec un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/
iam-reference/security-iam-awsmanpol.adoc.atom
```

Modification	Description	Date
Autorisation ajoutée pour the section called "AWS politique gérée : Amazon EKSLocal OutpostServiceRolePolicy"	`ssmmessages:OpenDataChannel` Autorisation ajoutée pour `AmazonEKSLocalOutpostServiceRolePolicy` .	26 juin 2025
Autorisation ajoutée pourthe section called "AWS politique gérée : Amazon EKSCompute Policy".	Autorisations de ressources mises à jour pour les `ec2:CreateFleet` actions `ec2:RunInstances` et afin d'inclure les réservations de capacité `arn:aws:ec2:*:*:capacity-reservation/*` . Cela permet au mode automatique d'Amazon EKS de lancer des instances en utilisant les réservations de capacité EC2 à la demande de votre compte. Ajouté `iam:CreateServiceLinkedRole` pour permettre au mode automatique d'Amazon EKS de créer le rôle lié au service EC2 Spot en votre `AWSServiceRoleForEC2Spot` nom.	20 juin 2025
Autorisation ajoutée à Amazon EKSService RolePolicy.	Ajout de `ec2:DescribeCapacityReservations` l'autorisation permettant au mode automatique d'Amazon EKS de lancer des instances en utilisant les réservations de capacité	20 juin 2025

Modification	Description	Date
	EC2 à la demande de votre compte.	
Présentéthe section called "AWS politique gérée : Amazon EKSDashboard ConsoleReadOnly".	A introduit une nouvelle `AmazonEKSDashboardConsoleReadOnly` politique.	19 juin 2025
Présentéthe section called "AWS politique gérée : Amazon EKSDashboard ServiceRolePolicy".	A introduit une nouvelle `AmazonEKSDashboardServiceRolePolicy` politique.	21 mai 2025
Des autorisations ont été ajoutées à Amazon EKSCluster Policy.	Ajout d'une `ec2:DeleteNetworkInterfaces` autorisation permettant à Amazon EKS de supprimer les interfaces réseau élastiques qui restent en place en cas de fermeture inattendue du VPC CNI.	16 avril 2025
Autorisation ajoutée à Amazon EKSService RolePolicy.	Ajout `ec2:RevokeSecurityGroupEgress` d'`ec2:AuthorizeSecurityGroupEgress` autorisations permettant aux AI/ML clients d'EKS d'ajouter des règles de sortie du groupe de sécurité au cluster EKS SG par défaut compatibles avec EFA, dans le cadre de la version 1.33 d'EKS.	14 avril 2025

Modification	Description	Date
Des autorisations ont été ajoutées à Amazon EKSService RolePolicy.	Ajout de l'autorisation de mettre fin aux EC2 instances créées par le mode automatique EKS.	28 février 2025
Des autorisations ont été ajoutées à Amazon EBSCSIDriver Policy.	Ajout d'une nouvelle déclaration autorisant le pilote EBS CSI à restaurer tous les instantanés. Cela était auparavant autorisé par la politique existante, mais une nouvelle déclaration explicite est requise en raison d'une modification de la gestion d'IAM pour`CreateVolume` . Ajout de la possibilité pour le pilote EBS CSI de modifier les balises sur les volumes existants. Le pilote EBS CSI peut modifier les balises des volumes existants via un paramètre dans Kubernetes. VolumeAttributesClasses Ajout de la possibilité pour le pilote EBS CSI d'activer la restauration rapide des instantanés (FSR) sur les volumes EBS. Le pilote EBS CSI peut activer le FSR sur de nouveaux volumes via les paramètres des classes de stockage Kubernetes.	13 janvier 2025

Modification	Description	Date
Autorisations ajoutées àthe section called "AWS politique gérée : Amazon EKSLoad BalancingPolicy".	Mis `AmazonEKSLoadBalancingPolicy` à jour pour permettre de répertorier et de décrire les ressources réseau et d'adresses IP.	26 décembre 2024
Autorisations ajoutées àthe section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegroup".	Mis à jour `AWSServiceRoleForAmazonEKSNodegroup` pour assurer la compatibilité avec les régions de Chine.	22 novembre 2024
Autorisations ajoutées à the section called "AWS politique gérée : Amazon EKSLocal OutpostClusterPolicy"	`ec2:DescribeAvailabilityZones` Autorisation ajoutée `AmazonEKSLocalOutpostClusterPolicy` afin que le AWS Cloud Controller Manager du plan de contrôle du cluster puisse identifier la zone de disponibilité dans laquelle se trouve chaque nœud.	21 novembre 2024
Autorisations ajoutées àthe section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegroup".	`AWSServiceRoleForAmazonEKSNodegroup` Politique mise à jour `ec2:RebootInstances` pour autoriser les instances créées par des groupes de nœuds gérés par Amazon EKS. Limite les `ec2:CreateTags` autorisations pour les EC2 ressources Amazon.	20 novembre 2024

Modification	Description	Date
Autorisations ajoutées àthe section called "AWS politique gérée : Amazon EKSService RolePolicy".	Politique AWS gérée mise à jour par EKS`AmazonEKSServiceRolePolicy`. Ajout d'autorisations pour les politiques d'accès EKS, la gestion de l'équilibreur de charge et le nettoyage automatique des ressources du cluster.	16 novembre 2024
Présentéthe section called "AWS politique gérée : Amazon EKSCompute Policy".	Politique AWS gérée mise à jour par EKS`AmazonEKSComputePolicy`. Autorisations de ressources mises à jour pour l'`iam:AddRoleToInstanceProfile` action.	7 novembre 2024
Présentéthe section called "AWS politique gérée : Amazon EKSCompute Policy".	AWS a présenté le`AmazonEKSComputePolicy`.	1er novembre 2024
Autorisations ajoutées à `AmazonEKSClusterPolicy`	Ajout `ec2:DescribeInstanceTopology` d'une autorisation permettant à Amazon EKS de joindre des informations topologiques au nœud sous forme d'étiquettes.	1er novembre 2024
Présentéthe section called "AWS politique gérée : Amazon EKSBlock StoragePolicy".	AWS a présenté le`AmazonEKSBlockStoragePolicy`.	30 octobre 2024

Modification	Description	Date
Présentéthe section called "AWS politique gérée : Amazon EKSLoad Balancing Policy".	AWS a présenté le`AmazonEKSLoadBalancingPolicy` .	30 octobre 2024
Des autorisations ont été ajoutées à Amazon EKSService RolePolicy.	Des `cloudwatch:PutMetricData` autorisations ont été ajoutées pour permettre à Amazon EKS de publier des métriques sur Amazon CloudWatch.	29 octobre 2024
Présentéthe section called "AWS politique gérée : Amazon EKSNetworking Policy".	AWS a présenté le`AmazonEKSNetworkingPolicy` .	28 octobre 2024
Autorisations ajoutées pour `AmazonEKSServicePolicy` et `AmazonEKSServiceRolePolicy`	Autorisations de balises ajoutées `ec2:GetSecurityGroupsForVpc` et associées pour permettre à EKS de lire les informations du groupe de sécurité et de mettre à jour les balises associées.	10 octobre 2024
Présentation d'Amazon EKSWorker NodeMinimalPolicy.	AWS a présenté le`AmazonEKSWorkerNodeMinimalPolicy` .	3 octobre 2024

Modification	Description	Date
Autorisations ajoutées à AWSServiceRoleForAmazonEKSNodegroup.	Ajout `autoscaling:ResumeProcesses` d'`autoscaling:SuspendProcesses` autorisations permettant à Amazon EKS de suspendre et de reprendre ses activités `AZRebalance` dans les groupes Auto Scaling gérés par Amazon EKS.	21 août 2024
Autorisations ajoutées à AWSServiceRoleForAmazonEKSNodegroup.	Ajout de `ec2:DescribeCapacityReservations` l'autorisation permettant à Amazon EKS de décrire la réservation de capacité dans le compte de l'utilisateur. `autoscaling:PutScheduledUpdateGroupAction` Autorisation ajoutée pour activer la configuration du dimensionnement planifié sur les groupes de `CAPACITY_BLOCK` nœuds.	27 juin 2024

Modification	Description	Date
Amazoneks_CNI_Policy — Mise à jour d'une politique existante	Amazon EKS a ajouté de nouvelles `ec2:DescribeSubnets` autorisations pour permettre au plugin Amazon VPC CNI pour Kubernetes de voir le nombre d'adresses IP gratuites dans vos sous-réseaux Amazon VPC. Le VPC CNI peut utiliser les adresses IP libres de chaque sous-réseau pour sélectionner les sous-réseaux avec le plus grand nombre d'adresses IP libres à utiliser lors de la création d'une interface elastic network.	4 mars 2024
Amazon EKSWorkerNodePolicy — Mise à jour d'une politique existante	Amazon EKS a ajouté de nouvelles autorisations pour permettre les identités du pod EKS. L'agent d'identité du pod Amazon EKS utilise le rôle de nœud.	26 novembre 2023
Présentation de la EFSCSIDriverpolitique d'Amazon.	AWS a présenté le`AmazonEFSCSIDriverPolicy` .	26 juillet 2023

Modification	Description	Date
Des autorisations ont été ajoutées à Amazon EKSCluster Policy.	Ajout d'une autorisation `ec2:DescribeAvailabilityZones` permettant à Amazon EKS d'obtenir les détails des zones de disponibilité lors de la découverte automatique des sous-réseaux lors de la création d'équilibreurs de charge.	7 février 2023
Conditions de politique mises à jour dans Amazon EBSCSIDriver Policy.	Conditions de stratégie non valides supprimées comportant des caractères génériques dans le champ clé `StringLike` . Nouvelle condition `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` ajoutée à `ec2:DeleteVolume` , permettant au pilote EBS CSI de supprimer les volumes créés par le plug-in intégré à l'arborescence.	17 novembre 2022

Modification	Description	Date
Des autorisations ont été ajoutées à Amazon EKSLocal OutpostServiceRolePolicy.	Ajout de `ec2:DescribeVPCAttribute`, `ec2:GetConsoleOutput` et `ec2:DescribeSecret` pour permettre une meilleure validation des prérequis et un meilleur contrôle du cycle de vie géré. Également ajoutés `ec2:DescribePlacementGroups` `"arn:aws:ec2:*:*:placement-group/*"` pour `ec2:RunInstances` prendre en charge le contrôle du placement des EC2 instances Amazon du plan de contrôle sur Outposts.	24 octobre 2022
Mettez à jour les autorisations d'Amazon Elastic Container Registry sur Amazon EKSLocal OutpostClusterPolicy.	Déplacement de l'action `ecr:GetDownloadUrlForLayer` de toutes les sections de ressources vers une section délimitée. Ajout de la ressource `arn:aws:ecr:*:*:repository/eks/`. Suppression de la ressource `arn:aws: ecr:`. Cette ressource est couverte par la ressource `arn:aws:ecr:*:*:repository/eks/*` ajoutée.	20 octobre 2022

Modification	Description	Date
Des autorisations ont été ajoutées à Amazon EKSLocal OutpostClusterPolicy.	Ajout du référentiel Amazon Elastic Container Registry `arn:aws: ecr:*:*:repository/kubelet-config-updater` pour que les instances du plan de contrôle du cluster puissent mettre à jour des arguments `kubelet`.	31 août 2022
Présentation d'Amazon EKSLocal OutpostClusterPolicy.	AWS a présenté le`AmazonEKSLocalOutpostClusterPolicy` .	24 août 2022
Présentation d'Amazon EKSLocal OutpostServiceRole Policy.	AWS a présenté le`AmazonEKSLocalOutpostServiceRolePolicy` .	23 août 2022
Présentation de la EBSCSIDriverpolitique d'Amazon.	AWS a présenté le`AmazonEBSCSIDriverPolicy` .	4 avril 2022
Des autorisations ont été ajoutées à Amazon EKSWorker NodePolicy.	Ajouté `ec2:DescribeInstanceTypes` pour permettre à Amazon EKS d'être optimisé pour détecter automatiquement AMIs les propriétés au niveau de l'instance.	21 mars 2022

Modification	Description	Date
Autorisations ajoutées à AWSServiceRoleForAmazonEKSNodegroup.	Ajout d'autorisations `autoscaling:EnableMetricsCollection` permettant à Amazon EKS d'activer la collecte de métriques.	13 décembre 2021
Des autorisations ont été ajoutées à Amazon EKSCluster Policy.	Ajout de `ec2:DescribeAccountAttributes` , `ec2:DescribeAddresses` et `ec2:DescribeInternetGateways` pour autoriser Amazon EKS à créer un rôle lié à un service pour un Network Load Balancer.	17 juin 2021
Amazon EKS a commencé à assurer le suivi des modifications.	Amazon EKS a commencé à suivre les modifications apportées AWS à ses politiques gérées.	17 juin 2021

Dépannage IAM

Cette rubrique traite de certaines erreurs courantes que vous pouvez rencontrer lorsque vous utilisez Amazon EKS avec IAM, ainsi que des solutions.

AccessDeniedException

Si vous recevez un message `AccessDeniedException` lors de l'appel d'une opération d' AWS API, cela signifie que les informations d'identification principales IAM que vous utilisez ne disposent pas des autorisations requises pour effectuer cet appel.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws: iam::111122223333:user/user_name is not authorized to perform:
```

```
eks:DescribeCluster on resource: arn:aws: eks:region:111122223333:cluster/my-cluster
```

Dans l'exemple de message ci-précédent, l'utilisateur n'est pas autorisé à appeler l'opération d'API `DescribeCluster` Amazon EKS. Pour accorder des autorisations d'administrateur Amazon EKS à un principal IAM, consultez la rubrique the section called "Politiques basées sur l'identité".

Pour plus d'informations générales sur IAM, consultez Contrôle de l'accès à l'aide de politiques dans le Guide de l'utilisateur IAM.

Impossible de voir les nœuds dans l'onglet Calcul ou dans l'onglet Ressources et vous recevez un message d'erreur dans AWS Management Console

Vous pouvez voir un message d'erreur de la console indiquant `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Assurez-vous que l'utilisateur principal IAM AWS Management Console avec lequel vous utilisez dispose des autorisations nécessaires. Pour de plus amples informations, veuillez consulter the section called "Autorisations requises".

aws-auth **`ConfigMap`** n'accorde pas l'accès au cluster

L'authentificateur AWS IAM n'autorise pas de chemin dans le rôle ARN utilisé dans le. `ConfigMap` Par conséquent, avant de spécifier `rolearn`, supprimez le chemin d'accès. Remplacez, par exemple, `arn:aws: iam::111122223333:role/team/developers/eks-admin` par `arn:aws: iam::111122223333:role/eks-admin` .

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon EKS.

Certains AWS services vous permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon EKS. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'effectuer l'`iam:PassRole`action.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Amazon EKS

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon EKS est compatible avec ces fonctionnalités, consultez the section called "Amazon EKS et IAM".
- Pour savoir comment fournir un accès à vos ressources sur les AWS comptes que vous possédez, consultez la section Fournir un accès à un utilisateur IAM sur un autre AWS compte que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des AWS comptes tiers, consultez la section Fournir un accès aux AWS comptes détenus par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Les conteneurs Pod reçoivent l'erreur suivante : **`An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`**

Vos conteneurs reçoivent cette erreur si votre application envoie explicitement des demandes au point de terminaison global AWS STS (`https://sts.amazonaws`) et si votre compte de service Kubernetes est configuré pour utiliser un point de terminaison régional. Vous pouvez résoudre le problème avec l'une des options suivantes :

- Mettez à jour le code de votre application pour supprimer les appels explicites au point de terminaison global AWS STS.
- Mettez à jour le code de votre application pour effectuer des appels explicites vers des points de terminaison régionaux tels que `https://sts.us-west-2.amazonaws.com` . Votre application doit comporter une redondance intégrée pour choisir une autre AWS région en cas de panne du service dans la AWS région. Pour plus d'informations, consultez la section Gestion du AWS STS dans une AWS région dans le guide de l'utilisateur IAM.
- Configurez vos comptes de service pour utiliser le point de terminaison mondial. Les clusters utilisent le point de terminaison régional par défaut. Pour de plus amples informations, veuillez consulter the section called "Points de terminaison STS".

Rôle IAM de cluster Amazon EKS

Un rôle IAM de cluster Amazon EKS est requis pour chaque cluster. Les clusters Kubernetes gérés par Amazon EKS utilisent ce rôle pour gérer les nœuds et l'ancien fournisseur de cloud utilise ce rôle pour créer des équilibreurs de charge avec Elastic Load Balancing for services.

Avant de pouvoir créer des clusters Amazon EKS, vous devez créer un rôle IAM avec l'une des politiques IAM suivantes :

- EKSClusterPolitique d'Amazon
- Une politique IAM personnalisée. Les autorisations minimales qui suivent permettent au cluster Kubernetes de gérer les nœuds, mais n'autorisent pas l'ancien fournisseur de cloud à créer des équilibreurs de charge avec Elastic Load Balancing. Votre politique IAM personnalisée doit disposer au moins des autorisations suivantes :

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws: ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTopology",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Avant le 3 octobre 2023, Amazon EKSCluster Policy était obligatoire concernant le rôle IAM de chaque cluster.
> Avant le 16 avril 2020, Amazon EKSService Policy et Amazon EKSCluster Policy étaient obligatoires et le nom suggéré pour le rôle était`eksServiceRole`. Avec le rôle `AWSServiceRoleForAmazonEKS` lié au service, la EKSServicepolitique Amazon Policy n'est plus requise pour les clusters créés le 16 avril 2020 ou après cette date.

Recherche d'un rôle de cluster existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de cluster Amazon EKS.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `eksClusterRole`. Si un rôle incluant `eksClusterRole` n'existe pas, reportez-vous the section called "Création du rôle de cluster Amazon EKS" à la section Création du rôle. Si un rôle qui inclut `eksClusterRole` existe, sélectionnez le rôle pour afficher les politiques attachées.
4. Choisissez Permissions (Autorisations).
5. Assurez-vous que la EKSClusterpolitique gérée par Amazon Policy est attachée au rôle. Si la politique est attachée, votre rôle de cluster Amazon EKS est configuré correctement.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation de confiance ne correspond pas, copiez la politique dans la fenêtre Modifier la politique de confiance et choisissez Mettre à jour la politique.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Création du rôle de cluster Amazon EKS

Vous pouvez utiliser la AWS Management Console ou la AWS CLI pour créer le rôle de cluster.

AWS Management Console

a. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
b. Choisissez Roles (Rôles) puis Create role (Créer un rôle).
c. Sous Type d'entité de confiance, sélectionnez AWS service.
d. Dans la liste déroulante Cas d'utilisation d'autres AWS services, sélectionnez EKS.
e. Sélectionnez EKS - Cluster pour votre cas d'utilisation, puis Next (Suivant).
f. Sous l'onglet Add permissions (Ajouter des autorisations), sélectionnez Next (Suivant).
g. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `eksClusterRole`.
h. Pour Description, saisissez un texte descriptif tel que `Amazon EKS - Cluster role`.
i. Sélectionnez Créer un rôle.

AWS CLI

a. Copiez le contenu suivant dans un fichier nommé *`cluster-trust-policy.json`*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez le rôle. Vous pouvez remplacer *`eksClusterRole`* par n'importe quel nom que vous choisissez.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

c. Attachez la politique IAM requise au rôle.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
```

```
  --role-name eksClusterRole
```

Rôle IAM de nœud Amazon EKS

Le `kubelet` démon du nœud Amazon EKS effectue des appels AWS APIs en votre nom. Les nœuds reçoivent l'autorisation pour ces appels d'API via un profil d'instance IAM et les politiques associées. Avant de pouvoir lancer les nœuds et les enregistrer dans un cluster, vous devez créer un rôle IAM qui sera utilisé par ces nœuds lors de leur lancement. Cette exigence s'applique aux nœuds lancés avec l'AMI optimisée Amazon EKS fournie par Amazon, ou avec tout autre nœud AMIs que vous avez l'intention d'utiliser. En outre, cette exigence s'applique à la fois aux groupes de nœuds gérés et aux nœuds autogérés.

> **Note**
>
> Vous ne pouvez pas utiliser le même rôle que celui utilisé pour créer des clusters.

Avant de créer des nœuds, vous devez créer un rôle IAM avec les autorisations suivantes :

- Autorisations permettant `kubelet` de décrire les EC2 ressources Amazon dans le VPC, telles que prévues par la politique d'Amazon EKSWorker NodePolicy. Cette politique fournit également les autorisations pour l'agent d'identité du pod Amazon EKS.
- Autorisations relatives `kubelet` à l'utilisation d'images de conteneurs issues d'Amazon Elastic Container Registry (Amazon ECR), telles que prévues par la politique d'Amazon. EC2 ContainerRegistryPullOnly Les autorisations d'utiliser des images de conteneurs provenant d'Amazon Elastic Container Registry (Amazon ECR) sont nécessaires parce que les modules complémentaires intégrés pour la mise en réseau exécutent des pods qui utilisent des images de conteneurs provenant d'Amazon ECR.
- (Facultatif) Autorisations permettant à l'agent d'identité du pod Amazon EKS d'utiliser l'action `eks-auth:AssumeRoleForPodIdentity` pour récupérer les informations d'identification pour les pods. Si vous n'utilisez pas Amazon EKSWorker NodePolicy, vous devez fournir cette autorisation en plus des EC2 autorisations d'utilisation d'EKS Pod Identity.
- (Facultatif) Si vous n'utilisez pas IRSA ou EKS Pod Identity pour accorder des autorisations aux pods VPC CNI, vous devez fournir des autorisations pour le VPC CNI sur le rôle d'instance. Vous pouvez utiliser la politique `AmazonEKS_CNI_Policy`gérée (si vous avez créé votre cluster avec la `IPv4` famille) ou une IPv6 politique que vous créez (si vous avez créé votre cluster avec la

IPv6 famille). Cependant, plutôt que d'attacher la politique à ce rôle, nous vous recommandons de l'attacher à un rôle distinct utilisé spécifiquement pour le module complémentaire Amazon VPC CNI. Pour plus d'informations sur la création d'un rôle distinct pour le module complémentaire Amazon VPC CNI, veuillez consulter the section called "Configuration pour IRSA".

> **Note**
>
> Les groupes de EC2 nœuds Amazon doivent avoir un rôle IAM différent de celui du profil Fargate. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM d'exécution de pod".

Recherche d'un rôle de nœud existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de nœud Amazon EKS.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole` dans la liste des rôles. Si aucun rôle portant l'un de ces noms n'existe, reportez-vous the section called "Création du rôle IAM de nœud Amazon EKS" à la section Création du rôle. Si un rôle qui contient `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole` existe, sélectionnez-le pour afficher les politiques attachées.
4. Choisissez Autorisations.
5. Assurez-vous que les politiques EC2 ContainerRegistryPullOnly gérées par Amazon EKSWorker NodePolicy et Amazon sont associées au rôle ou qu'une politique personnalisée est attachée avec les autorisations minimales.

   > **Note**
   >
   > Si la politique AmazonEKS_CNI_Policy est attachée au rôle, nous vous recommandons de la supprimer et de l'attacher à un rôle IAM qui est plutôt mappé au compte de service `aws-node` Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation de confiance ne correspond pas, copiez la politique dans la fenêtre Modifier la politique de confiance et choisissez Mettre à jour la politique.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Création du rôle IAM de nœud Amazon EKS

Vous pouvez créer le rôle IAM du nœud à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

a. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
b. Dans le volet de navigation de gauche, choisissez Rôles.
c. Sur la page Rôles, choisissez Créer un rôle.
d. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   i. Dans la section Type d'entité fiable, sélectionnez AWS service.
   ii. Sous Use case (Cas d'utilisation), choisissez EC2.
   iii. Choisissez Suivant.

e. Sur la page Ajouter des autorisations, associez une stratégie personnalisée ou procédez comme suit :

   i. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEKSWorkerNodePolicy`.

   ii. Cochez la case située à gauche d'Amazon EKSWorker NodePolicy dans les résultats de recherche.

   iii. Sélectionnez Clear filters (Effacer les filtres).

   iv. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEC2ContainerRegistryPullOnly`.

   v. Cochez la case située à gauche d'Amazon EC2 ContainerRegistryPullOnly dans les résultats de recherche.

      La politique gérée Amazoneks_CNI_Policy ou une politique que vous créez doit également être attachée à ce rôle ou à IPv6 un autre rôle mappé au compte de service Kubernetes. `aws-node` Nous vous recommandons d'attribuer la politique au rôle associé au compte de service Kubernetes au lieu de l'affecter à ce rôle. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

   vi. Choisissez Suivant.

f. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   i. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSNodeRole`.

   ii. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Node role`.

   iii. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

   iv. Sélectionnez Créer un rôle.

AWS CLI

a. Exécutez la commande ci-dessous pour créer un fichier`node-role-trust-relationship.json`.

```
cat >node-role-trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
EOF
```

b. Créez le rôle IAM.

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

c. Attachez deux politiques gérées IAM requises au rôle IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly \
  --role-name AmazonEKSNodeRole
```

d. Attachez l'une des politiques IAM suivantes au rôle IAM en fonction de la famille IP avec laquelle vous avez créé votre cluster. La politique doit être attachée à ce rôle ou à un rôle associé au compte de `aws-node` service Kubernetes utilisé pour le plug-in Amazon VPC CNI pour Kubernetes. Nous vous recommandons d'attribuer la politique au rôle associé au compte de service Kubernetes. Pour attribuer la politique au rôle associé au compte de service Kubernetes, veuillez consulter the section called “Configuration pour IRSA”.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

- IPv6

  A. Copiez la politique suivante et enregistrez-la dans un fichier appelé `vpc-cni-ipv6-policy.json`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws: ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

  B. Créez la politique IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

  C. Attachez la politique IAM au rôle IAM. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Rôle IAM du cluster en mode automatique Amazon EKS

Un rôle IAM de cluster Amazon EKS est requis pour chaque cluster. Les clusters Kubernetes gérés par Amazon EKS utilisent ce rôle pour automatiser les tâches de routine liées au stockage, à la mise en réseau et au dimensionnement automatique du calcul.

Avant de créer des clusters Amazon EKS, vous devez créer un rôle IAM avec les politiques requises pour le mode automatique EKS. Vous pouvez soit joindre les stratégies gérées par AWS IAM suggérées, soit créer des politiques personnalisées avec des autorisations équivalentes.

- EKSComputePolitique d'Amazon
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- EKSNetworkingPolitique d'Amazon
- EKSClusterPolitique d'Amazon

Recherche d'un rôle de cluster existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de cluster Amazon EKS.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `AmazonEKSAutoClusterRole`. Si un rôle incluant `AmazonEKSAutoClusterRole` n'existe pas, consultez les instructions de la section suivante pour créer le rôle. Si un rôle qui inclut `AmazonEKSAutoClusterRole` existe, sélectionnez le rôle pour afficher les politiques attachées.
4. Choisissez Permissions (Autorisations).
5. Assurez-vous que la EKSClusterpolitique gérée par Amazon Policy est attachée au rôle. Si la politique est attachée, votre rôle de cluster Amazon EKS est configuré correctement.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation de confiance ne correspond pas, copiez la politique dans la fenêtre Modifier la politique de confiance et choisissez Mettre à jour la politique.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

> **Note**
>
> AWS ne nécessite pas le nom `AmazonEKSAutoClusterRole` de ce rôle.

Création du rôle de cluster Amazon EKS

Vous pouvez utiliser la AWS Management Console ou la AWS CLI pour créer le rôle de cluster.

AWS Management Console

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Roles (Rôles) puis Create role (Créer un rôle).
3. Sous Type d'entité de confiance, sélectionnez AWS service.
4. Dans la liste déroulante Cas d'utilisation pour d'autres AWS services, sélectionnez EKS.
5. Sélectionnez EKS - Cluster pour votre cas d'utilisation, puis Next (Suivant).
6. Dans l'onglet Ajouter des autorisations, sélectionnez les politiques, puis cliquez sur Suivant.
   - EKSComputePolitique d'Amazon
   - AmazonEKSBlockStoragePolicy
   - AmazonEKSLoadBalancingPolicy

- EKSNetworkingPolitique d'Amazon
- EKSClusterPolitique d'Amazon

7. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSAutoClusterRole`.
8. Pour Description, saisissez un texte descriptif tel que `Amazon EKS - Cluster role`.
9. Choisissez Créer un rôle.

AWS CLI

1. Copiez le contenu suivant dans un fichier nommé *`cluster-trust-policy.json`*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Créez le rôle. Vous pouvez remplacer *`AmazonEKSAutoClusterRole`* par n'importe quel nom que vous choisissez.

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Associez les politiques IAM requises au rôle :

EKSClusterPolitique d'Amazon :

```
aws iam attach-role-policy \
```

```
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputePolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBlock StoragePolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

Amazon EKSLoad BalancingPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Rôle IAM du nœud Amazon EKS Auto Mode

> **Note**
>
> Vous ne pouvez pas utiliser le même rôle que celui utilisé pour créer des clusters.

Avant de créer des nœuds, vous devez créer un rôle IAM avec les politiques suivantes, ou des autorisations équivalentes :

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Recherche d'un rôle de nœud existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de nœud Amazon EKS.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `AmazonEKSAutoNodeRole`. Si aucun rôle portant l'un de ces noms n'existe, consultez les instructions de la section suivante pour créer le rôle. Si un rôle qui contient `AmazonEKSAutoNodeRole` existe, sélectionnez le rôle pour afficher les stratégies attachées.
4. Choisissez Autorisations.
5. Assurez-vous que les politiques requises ci-dessus sont jointes, ou des politiques personnalisées équivalentes.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique de confiance et choisissez Mettre à jour la politique.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Création du rôle IAM de nœud Amazon EKS

Vous pouvez créer le rôle IAM du nœud à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Dans la section Type d'entité fiable, sélectionnez AWS service.
   b. Sous Use case (Cas d'utilisation), choisissez EC2.
   c. Choisissez Suivant.
5. Sur la page Ajouter des autorisations, joignez les politiques suivantes :
   - AmazonEKSWorkerNodeMinimalPolicy
   - AmazonEC2ContainerRegistryPullOnly
6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :
   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSAutoNodeRole`.
   b. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Node role`.
   c. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
   d. Choisissez Créer un rôle.

AWS CLI

Création du rôle Node IAM

Utilisez le fichier node-trust-policy.json de l'étape précédente pour définir les entités qui peuvent assumer le rôle. Exécutez la commande suivante pour créer le rôle Node IAM :

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Notez l'ARN du rôle

Après avoir créé le rôle, récupérez et enregistrez l'ARN du rôle Node IAM. Vous aurez besoin de cet ARN dans les étapes suivantes. Utilisez la commande suivante pour obtenir l'ARN :

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Joindre les politiques requises

Associez les politiques AWS gérées suivantes au rôle Node IAM pour fournir les autorisations nécessaires :

Pour joindre Amazon, procédez comme suit EKSWorker NodeMinimalPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Pour joindre Amazon, procédez comme suit EC2 ContainerRegistryPullOnly :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Surveillez les performances de votre cluster et consultez les journaux

Vous pouvez observer vos données dans Amazon EKS à l'aide de nombreux outils de surveillance ou de journalisation disponibles. Les données de vos journaux Amazon EKS peuvent être transmises à AWS des services ou à des outils partenaires pour l'analyse des données. De nombreux services sont disponibles dans le document AWS Management Console qui fournissent des données pour résoudre vos problèmes avec Amazon EKS. Vous pouvez également utiliser une solution open source AWS prise en charge pour surveiller l'infrastructure Amazon EKS.

Après avoir sélectionné Clusters dans le volet de navigation gauche de la console Amazon EKS, vous pouvez consulter l'état et les détails du cluster en choisissant le nom de votre cluster et en choisissant l'onglet Observabilité. Pour afficher des détails sur les ressources Kubernetes existantes déployées sur votre cluster, consultez the section called "Accès aux ressources du cluster".

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon EKS et de vos AWS solutions. Nous vous recommandons de collecter des données de surveillance à partir de toutes les parties de votre AWS solution. De cette façon, vous pouvez déboguer plus facilement une éventuelle défaillance à plusieurs points. Avant de commencer à surveiller Amazon EKS, assurez-vous que votre plan de surveillance réponde aux questions suivantes.

- Quels sont vos objectifs ? Avez-vous besoin de notifications en temps réel si vos clusters évoluent considérablement ?
- Quelles ressources faut-il observer ?
- À quelle fréquence devez-vous observer ces ressources ? Votre entreprise veut-elle réagir rapidement aux risques ?
- Quels outils avez-vous l'intention d'utiliser ? Si vous exécutez déjà AWS Fargate lors de votre lancement, vous pouvez utiliser le routeur de log intégré.
- Qui prévoyez-vous d'effectuer les tâches de surveillance ?
- À qui voulez-vous que les notifications soient envoyées en cas de problème ?

Surveillance et journalisation sur Amazon EKS

Amazon EKS fournit des outils intégrés de surveillance et de journalisation. Pour les versions prises en charge, le tableau de bord d'observabilité donne une visibilité sur les performances de votre cluster. Il vous permet de détecter, de résoudre et de résoudre rapidement les problèmes. Outre les fonctionnalités de surveillance, il inclut des listes basées sur les journaux d'audit du plan de contrôle. Le plan de contrôle Kubernetes expose un certain nombre de métriques qui peuvent également être supprimées en dehors de la console.

La journalisation du plan de contrôle enregistre tous les appels d'API vers vos clusters, les informations d'audit capturant quels utilisateurs ont effectué quelles actions sur vos clusters, et les informations basées sur les rôles. Pour plus d'informations, consultez la section Journalisation et surveillance sur Amazon EKS dans le guide AWS prescriptif.

La journalisation du plan de contrôle Amazon EKS fournit des journaux d'audit et de diagnostic directement depuis le plan de contrôle Amazon EKS vers CloudWatch les journaux de votre compte. Ces journaux vous permettent de sécuriser et d'exécuter facilement vos clusters. Vous pouvez sélectionner les types de journaux exacts dont vous avez besoin, et les journaux sont envoyés sous forme de flux de journaux à un groupe pour chaque cluster Amazon EKS inclus CloudWatch. Pour de plus amples informations, veuillez consulter the section called “Journaux du plan de contrôle”.

> **Note**
>
> Lorsque vous consultez les journaux de l'authentificateur Amazon EKS sur Amazon CloudWatch, les entrées affichées contiennent du texte similaire à l'exemple de texte suivant.
>
> ```
> level=info msg="mapping IAM role" groups="[]" role="arn:aws:
> iam::111122223333:role/XXXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXXX"
> username="eks:node-manager"
> ```
>
> Des entrées contenant ce texte sont attendues. `username` est un rôle de service interne Amazon EKS qui effectue des opérations spécifiques pour les groupes de nœuds gérés et Fargate.
> Pour une journalisation personnalisable de bas niveau, la journalisation Kubernetes est disponible.

Amazon EKS est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon EKS. CloudTrail capture tous les appels d'API pour Amazon EKS sous forme d'événements. Ces captures incluent les appels de la console Amazon EKS et les appels de code vers les opérations d'API Amazon EKS. Pour de plus amples informations, veuillez consulter the section called " AWS CloudTrail".

Le serveur d'API Kubernetes expose un certain nombre de métriques qui sont utiles pour la surveillance et l'analyse. Pour de plus amples informations, veuillez consulter the section called "Métriques Prometheus".

Pour configurer Fluent Bit pour les CloudWatch journaux Amazon personnalisés, consultez la section Configuration de Fluent Bit dans le guide de CloudWatch l'utilisateur Amazon.

Outils de surveillance et de journalisation Amazon EKS

Amazon Web Services fournit différents outils que vous pouvez utiliser pour surveiller Amazon EKS. Vous pouvez configurer certains outils pour configurer la surveillance automatique, mais certains nécessitent des appels manuels. Nous vous recommandons d'automatiser les tâches de surveillance autant que votre environnement et le jeu d'outils existant le permettent.

Le tableau suivant décrit les différentes options de l'outil de surveillance.

Zones	Outil	Description	Configuration
Plan de contrôle	Tableau de bord d'observabilité	Pour les versions prises en charge, le tableau de bord d'observabilité donne une visibilité sur les performances de votre cluster. Il vous permet de détecter, de résoudre et de résoudre rapidement les problèmes.	Procédures de configuration
Applications/plan de commande	Prometheus	Prometheus peut être utilisé pour surveille	Procédures de configuration

Zones	Outil	Description	Configuration
		r les métriques et les alertes relatives aux applications et au plan de contrôle.	
Applications	CloudWatch Container Insights	CloudWatch Container Insights collecte, agrège et résume les métriques et les journaux de vos applications conteneurisées et de vos microservices.	Procédures de configuration
Applications	AWS Distribution pour OpenTelemetry (ADOT)	ADOT peut collecter et envoyer des métriques corrélées, des données de suivi et des métadonné es aux services de AWS surveillance ou aux partenaires. Il peut être configuré via CloudWatch Container Insights.	Procédures de configuration
Applications	Amazon DevOps Guru	Amazon DevOps Guru détecte les performances opérationnelles et la disponibilité au niveau des nœuds.	Procédures de configuration

Zones	Outil	Description	Configuration
Applications	AWS X-Ray	AWS X-Ray reçoit des données de suivi concernant votre application. Ces données de suivi incluent les demandes entrantes et sortantes ainsi que les métadonné es relatives aux demandes. Pour Amazon EKS, l'implémentation nécessite le OpenTelemetry module complémen taire.	Procédures de configuration
Applications	Amazon CloudWatch	CloudWatch fournit gratuitement certaines métriques de base d'Amazon EKS sur les versions prises en charge. Vous pouvez étendre cette fonctionnalité avec l'opérateur CloudWatc h d'observabilité pour gérer la collecte des métriques, des journaux et des données de suivi.	Procédures de configuration

Le tableau suivant décrit les différentes options de l'outil de journalisation.

Zones	Outil	Description	Configuration
Plan de contrôle	Tableau de bord d'observabilité	Pour les versions prises en charge, le tableau de bord d'observabilité affiche des listes basées sur les journaux d'audit du plan de contrôle. Il inclut également des liens vers les journaux du plan de contrôle sur Amazon CloudWatch.	Procédures de configuration
Applications	Informations sur les CloudWatch conteneurs Amazon	Amazon CloudWatch Container Insights collecte, agrège et résume les métriques et les journaux provenant de vos applications conteneurisées et de vos microservices.	Procédures de configuration
Plan de contrôle	Amazon CloudWatch Logs	Vous pouvez envoyer des journaux d'audit et de diagnostic directement depuis le plan de contrôle Amazon EKS vers CloudWatch Logs in your account.	Procédures de configuration
Plan de contrôle	AWS CloudTrail	Il consigne les appels d'API par un utilisate	Procédures de configuration

Zones	Outil	Description	Configuration
		ur, un rôle ou un service.	
Plusieurs zones pour les instances AWS Fargate	AWS Routeur Fargate Log	Pour les instances AWS Fargate, le routeur de journaux transmet les journaux aux services ou aux AWS outils partenaires. Il est utilisé AWS pour Fluent Bit. Les journaux peuvent être diffusés vers d'autres AWS services ou outils partenaires.	Procédures de configuration

Surveillez votre cluster à l'aide du tableau de bord d'observabilité

La console Amazon EKS inclut un tableau de bord d'observabilité qui donne une visibilité sur les performances de votre cluster. Les informations qu'il fournit vous aident à détecter, résoudre et résoudre rapidement les problèmes. Vous pouvez ouvrir la section correspondante du tableau de bord d'observabilité en choisissant un élément dans le résumé de la santé et des performances. Ce résumé est inclus à plusieurs endroits, notamment dans l'onglet Observabilité.

Le tableau de bord d'observabilité est divisé en plusieurs onglets.

Récapitulatif

Le résumé de la santé et des performances répertorie la quantité d'articles dans différentes catégories. Chaque chiffre fait office de lien hypertexte vers un emplacement dans le tableau de bord d'observabilité avec une liste pour cette catégorie.

Santé du cluster

L'état du cluster fournit des notifications importantes à prendre en compte, dont vous devrez peut-être prendre des mesures dès que possible. Dans cette liste, vous pouvez consulter les descriptions et les

ressources concernées. L'état de santé du cluster comprend deux tableaux : Problèmes de santé et Informations sur la configuration. Pour actualiser l'état des problèmes de santé, cliquez sur le bouton d'actualisation (↻). Les informations de configuration sont mises à jour automatiquement toutes les 24 heures et ne peuvent pas être actualisées manuellement.

Pour plus d'informations sur les problèmes de santé, consultezthe section called "Codes d'erreur FAQs et d'intégrité du cluster avec chemins de résolution". Pour plus d'informations sur les informations de configuration, consultezthe section called "Informations sur le cluster".

Surveillance du plan de contrôle

L'onglet Surveillance du plan de contrôle est divisé en trois sections, chacune vous permettant de surveiller et de dépanner le plan de contrôle de votre cluster.

Métriques

Pour les clusters de versions `1.28` Kubernetes ou supérieures, la section Metrics présente des graphiques de plusieurs métriques collectées pour différents composants du plan de contrôle.

Vous pouvez définir la période utilisée par l'axe X de chaque graphique en effectuant des sélections en haut de la section. Vous pouvez actualiser les données à l'aide du bouton d'actualisation (↻). Pour chaque graphique distinct, le bouton représentant des ellipses verticales (1) ouvre un menu contenant les options de CloudWatch.

Ces métriques et bien d'autres sont automatiquement disponibles CloudWatch sous forme de métriques de surveillance de base dans l'espace de `AWS/EKS` noms. Pour plus d'informations, consultez les sections Surveillance de base et surveillance détaillée dans le guide de CloudWatch l'utilisateur Amazon. Pour obtenir des statistiques, des visualisations et des informations plus détaillées, consultez Container Insights dans le guide de CloudWatch l'utilisateur Amazon. Ou si vous préférez une surveillance basée sur Prometheus, consultez. the section called "Métriques Prometheus"

Le tableau suivant décrit les mesures disponibles.

Métrique	Description
APIServer Demandes	Les demandes par minute adressées au serveur API.

Métrique	Description
APIServer Nombre total de demandes : 4XX	Nombre de demandes de serveur d'API par minute contenant des codes de réponse HTTP 4XX (erreurs côté client).
APIServer Nombre total de demandes : 5XX	Nombre de demandes de serveur d'API par minute contenant des codes de réponse HTTP 5XX (erreurs côté serveur).
APIServer Nombre total de demandes 429	Nombre de demandes de serveur d'API par minute contenant des codes de réponse HTTP 429 (trop de demandes).
Taille de stockage	Taille de la base de données de stockage (`etcd`).
Tentatives du planificateur	Nombre de tentatives de planification des modules en fonction des résultats « non planifiables », « erreurs » et « planifiées ».
Pods en attente	Le nombre de modules en attente par type de file d'attente : « actif », « en attente », « non planifiable » et « fermé ».
Latence des demandes du serveur API	La latence des demandes du serveur d'API.
Demandes en vol en cours du serveur API	Les demandes en cours pour le serveur API en cours.
Demandes de webhook	Les demandes de webhook par minute.
Refus de demandes de webhook	Nombre de demandes de webhook rejetées.
Latence des requêtes Webhook P99	99e percentile de latence des demandes de webhook externes provenant de tiers.

CloudWatch Informations sur les journaux

La section CloudWatch Log Insights affiche différentes listes basées sur les journaux d'audit du plan de contrôle. Les journaux du plan de contrôle Amazon EKS doivent être activés pour utiliser cette fonctionnalité, ce que vous pouvez faire dans la section Afficher les journaux du plan de contrôle dans la CloudWatch section Afficher les journaux du plan de contrôle.

Lorsque suffisamment de temps s'est écoulé pour collecter les données, vous pouvez exécuter toutes les requêtes ou choisir Exécuter la requête pour une seule liste à la fois. Un coût supplémentaire sera facturé CloudWatch chaque fois que vous lancerez des requêtes. Choisissez la période des résultats que vous souhaitez afficher en haut de la section. Si vous souhaitez un contrôle plus avancé pour n'importe quelle requête, vous pouvez choisir Afficher dans CloudWatch. Cela vous permettra de mettre à jour une requête CloudWatch en fonction de vos besoins.

Pour plus d'informations, consultez Analyser les données des CloudWatch journaux avec Logs Insights dans le guide de l'utilisateur Amazon CloudWatch Logs.

Afficher les connexions au plan de contrôle CloudWatch

Choisissez Gérer la journalisation pour mettre à jour les types de journaux disponibles. Plusieurs minutes sont nécessaires pour que les journaux apparaissent dans CloudWatch les journaux une fois que vous avez activé la journalisation. Lorsque suffisamment de temps s'est écoulé, cliquez sur l'un des liens Afficher de cette section pour accéder au journal correspondant.

Pour de plus amples informations, veuillez consulter the section called “Journaux du plan de contrôle”.

Informations sur le cluster

Le tableau Upgrade Insights met en évidence les problèmes et recommande des mesures correctives, accélérant ainsi le processus de validation pour la mise à niveau vers les nouvelles versions de Kubernetes. Amazon EKS analyse automatiquement les clusters par rapport à une liste de problèmes potentiels liés à la mise à niveau de la version de Kubernetes. Le tableau Upgrade Insights répertorie les vérifications d'informations effectuées par Amazon EKS sur ce cluster, ainsi que leurs statuts associés.

Amazon EKS gère et actualise régulièrement la liste des vérifications d'informations à effectuer sur la base des évaluations des modifications apportées au projet Kubernetes ainsi que des modifications du service Amazon EKS liées aux nouvelles versions. La console Amazon EKS

actualise automatiquement le statut de chaque information, qui est visible dans la colonne de date de dernière actualisation.

Pour de plus amples informations, veuillez consulter the section called "Informations sur le cluster".

Problèmes de santé des nœuds

L'agent de surveillance des nœuds Amazon EKS lit automatiquement les journaux des nœuds pour détecter les problèmes de santé. Quel que soit le paramètre de réparation automatique, tous les problèmes de santé du nœud sont signalés afin que vous puissiez les examiner si nécessaire. Si un type de problème est répertorié sans description, vous pouvez lire la description dans son élément contextuel.

Lorsque vous actualisez la page, tous les problèmes résolus disparaissent de la liste. Si la réparation auto est activée, vous pourriez voir temporairement certains problèmes de santé qui seront résolus sans action de votre part. Les problèmes qui ne sont pas pris en charge par la réparation auto peuvent nécessiter une action manuelle de votre part, selon le type.

Pour que les problèmes de santé des nœuds soient signalés, votre cluster doit utiliser le mode automatique Amazon EKS ou disposer du module complémentaire d'agent de surveillance des nœuds. Pour de plus amples informations, veuillez consulter the section called "Santé des nœuds".

Surveillez les indicateurs de votre cluster avec Prometheus

Prometheus est une base de données de surveillance et de séries chronologiques qui analyse les terminaux. Elle permet d'interroger, d'agréger et de stocker les données collectées. Vous pouvez également l'utiliser pour les alertes et l'agrégation des alertes. Cette rubrique explique comment configurer Prometheus en tant qu'option gérée ou open source. La surveillance des métriques du plan de contrôle d'Amazon EKS est un cas d'utilisation courant.

Amazon Managed Service for Prometheus est un service de surveillance et d'alerte compatible avec Prometheus qui facilite la surveillance des applications et infrastructures conteneurisées à grande échelle. Il s'agit d'un service entièrement géré qui met automatiquement à l'échelle l'ingestion, le stockage, l'interrogation et l'alerte de vos métriques. Il s'intègre également aux services AWS de sécurité pour permettre un accès rapide et sécurisé à vos données. Vous pouvez utiliser le langage de requête open source PromQL pour interroger vos métriques et émettre des alertes à leur sujet. Vous pouvez également utiliser le gestionnaire d'alertes d'Amazon Managed Service for Prometheus pour configurer des règles d'alerte pour les alertes critiques. Vous pouvez ensuite envoyer ces alertes critiques sous forme de notifications à une rubrique Amazon SNS.

Il existe plusieurs options pour utiliser Prometheus avec Amazon EKS :

- Vous pouvez activer les métriques Prometheus lors de la première création d'un cluster Amazon EKS ou créer votre propre outil Prometheus Scraper pour les clusters existants. Ces deux options sont abordées dans cette rubrique.
- Vous pouvez déployer Prometheus à l'aide de Helm. Pour de plus amples informations, veuillez consulter the section called "Déployer avec Helm".
- Vous pouvez consulter les métriques brutes du plan de contrôle au format Prometheus. Pour de plus amples informations, veuillez consulter the section called "Plan de contrôle".

Étape 1 : activer les métriques Prometheus

> **Important**
>
> Les ressources Amazon Managed Service for Prometheus ne font pas partie du cycle de vie du cluster et doivent être gérées indépendamment du cluster. Lorsque vous supprimez votre cluster, assurez-vous de supprimer également tous les scrapers applicables afin de mettre fin aux coûts applicables. Pour plus d'informations, consultez la section Rechercher et supprimer des scrapers dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

Prometheus découvre et collecte les métriques de votre cluster grâce à un modèle basé sur l'extraction appelé scraping. Les scrapers sont configurés pour collecter des données à partir de l'infrastructure de votre cluster et des applications conteneurisées. Lorsque vous activez l'option d'envoi des métriques Prometheus, Amazon Managed Service for Prometheus fournit un scraper sans agent entièrement géré.

Si vous n'avez pas encore créé le cluster, vous pouvez activer l'option permettant d'envoyer des métriques à Prometheus lors de la création initiale du cluster. Dans la console Amazon EKS, cette option se trouve dans l'étape Configurer l'observabilité qui consiste à créer un nouveau cluster. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster ".

Si vous possédez déjà un cluster, vous pouvez créer votre propre grattoir Prometheus. Pour ce faire, dans la console Amazon EKS, accédez à l'onglet Observabilité de votre cluster et cliquez sur le bouton Ajouter un scraper. Si vous préférez utiliser l' AWS API ou la AWS CLI, consultez Create a scraper dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

Les options suivantes sont disponibles lors de la création du scraper avec la console Amazon EKS.

Alias de Scraper

(Facultatif) Saisissez un alias unique pour le scraper.

Destination (Destination)

Choisissez un espace de travail Amazon Managed Service for Prometheus. Un espace de travail est un espace logique dédié au stockage et à l'interrogation des métriques Prometheus. Grâce à cet espace de travail, vous pourrez consulter les statistiques de Prometheus sur tous les comptes qui y ont accès. L'option Créer un nouvel espace de travail indique à Amazon EKS de créer un espace de travail en votre nom à l'aide de l'Alias d'espace de travail que vous fournissez. L'option Sélectionner un espace de travail existant vous permet de sélectionner un espace de travail existant dans une liste déroulante. Pour plus d'informations sur les espaces de travail, consultez Gestion des espaces de travail dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.

Accès au service

Cette section résume les autorisations que vous accordez lors de l'envoi de métriques Prometheus :

- Autoriser Amazon Managed Service for Prometheus à décrire le cluster Amazon EKS récupéré
- Autoriser l'écriture à distance dans l'espace de travail Amazon Managed Prometheus

Si `AmazonManagedScraperRole` existe déjà, le scraper l'utilise. Choisissez le lien `AmazonManagedScraperRole` pour voir les Détails de l'autorisation. Si ce `AmazonManagedScraperRole` n'est pas déjà le cas, cliquez sur le lien Afficher les détails des autorisations pour voir les autorisations spécifiques que vous accordez en envoyant des métriques à Prometheus.

Sous-réseaux

Modifiez les sous-réseaux dont héritera le scraper selon vos besoins. Si vous devez ajouter une option de sous-réseau grisée, revenez à l'étape Créer un cluster Spécifier le réseau.

Configuration du grattoir

Modifiez la configuration du scraper au format YAML si nécessaire. Pour ce faire, utilisez le formulaire ou chargez un fichier YAML de remplacement. Pour plus d'informations, consultez Configuration du scraper dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.

Amazon Managed Service for Prometheus fait référence au scraper sans agent créé à côté du cluster en tant que collecteur géré. AWS Pour plus d'informations sur les collecteurs AWS gérés, consultez

la section Ingestion des métriques avec des collecteurs AWS gérés dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

> **Important**
>
> - Si vous créez un scraper Prometheus à l'aide de AWS la AWS CLI ou de l'API, vous devez ajuster sa configuration pour donner au scraper des autorisations au sein du cluster. Pour plus d'informations, consultez Configuration de votre cluster Amazon EKS dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.
> - Si vous avez un scraper Prometheus créé avant le 11 novembre 2024 qui utilise les entrées au lieu d'accès, vous devez `aws-auth` `ConfigMap` le mettre à jour pour accéder à des métriques supplémentaires depuis le plan de contrôle du cluster Amazon EKS. Pour la configuration mise à jour, consultez la section Configuration manuelle d'Amazon EKS pour l'accès au scraper dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

Étape 2 : Utiliser les métriques Prometheus

Pour plus d'informations sur l'utilisation des métriques Prometheus une fois que vous les avez activées pour votre cluster, consultez le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

Étape 3 : Gérez les grattoirs Prometheus

Pour gérer les scrapers, choisissez l'onglet Observability dans la console Amazon EKS. Un tableau affiche la liste des scrappers du cluster, avec des informations telles que l'ID de scraper, l'alias, le statut et la date de création. Vous pouvez ajouter d'autres grattoirs, modifier des grattoirs, supprimer des grattoirs ou afficher plus d'informations sur les grattoirs actuels.

Pour en savoir plus sur un grattoir, cliquez sur le lien « ID du grattoir ». Par exemple, vous pouvez consulter l'ARN, l'environnement, l'ID de l'espace de travail, le rôle IAM, la configuration et les informations réseau. Vous pouvez utiliser l'identifiant du scraper comme entrée dans Amazon Managed Service pour les opérations `DescribeScraper`d'API Prometheus, telles que, et. `UpdateScraperDeleteScraper` Pour plus d'informations sur l'utilisation de l'API Prometheus, consultez le manuel Amazon Managed Service for Prometheus API Reference.

Déployez Prometheus à l'aide de Helm

Au lieu d'utiliser Amazon Managed Service pour Prometheus, vous pouvez déployer Prometheus dans votre cluster avec Helm. Si Helm est déjà installé, vous pouvez vérifier votre version avec la commande `helm version`. Helm est un gestionnaire de package pour les clusters Kubernetes. Pour de plus amples informations sur Helm et son installation, veuillez consulter the section called "Déployez des applications avec Helm".

Une fois que vous avez configuré Helm pour votre cluster Amazon EKS, vous pouvez l'utiliser pour déployer Prometheus avec les étapes suivantes.

1. Créez un espace de noms Prometheus.

```
kubectl create namespace prometheus
```

2. Ajoutez le référentiel de graphiques `prometheus-community`.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Déployez Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> Si vous obtenez l'erreur `Error: failed to download "stable/prometheus" (hint: running helm repo update may help)` lors de l'exécution de cette commande, exécutez `helm repo update prometheus-community`, puis réessayez d'exécuter la commande Étape 2.

Si vous obtenez l'erreur `Error: rendered manifests contain a resource that already exists`, exécutez `helm uninstall` *`your-release-name`* `-n` *`namespace`* , puis réessayez d'exécuter la commande Étape 3.

4. Vérifiez que tous les pods de l'espace de `prometheus` noms sont en bon `READY` état.

```
kubectl get pods -n prometheus
```

L'exemple qui suit illustre un résultat.

```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

5. Utilisez `kubectl` pour transférer la console Prometheus vers votre ordinateur local.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Pointez un navigateur Web sur `http://localhost:9090` pour afficher la console Prometheus.
7. Choisissez une métrique dans le menu - insert metric at cursor (insérer une métrique sur le curseur), puis choisissez Execute (Exécuter). Choisissez l'onglet Graph (Graphique) pour afficher la métrique au fil du temps. L'image suivante montre `container_memory_usage_bytes` au fil du temps.

8. Dans la barre de navigation supérieure, choisissez Status (Statut), puis Targets (Cibles).

Tous les points de terminaison Kubernetes qui sont connectés à Prometheus à l'aide de la découverte de service sont affichés.

Récupère les métriques brutes du plan de contrôle au format Prometheus

Le plan de contrôle Kubernetes expose un certain nombre de métriques représentées au format Prometheus. Ces métriques sont utiles pour la surveillance et l'analyse. Ils sont exposés en interne via des points de terminaison de métriques et sont accessibles sans avoir à déployer complètement Prometheus. Cependant, déployer Prometheus plus facilement permet d'analyser les métriques au fil du temps.

Pour afficher la sortie des métriques brutes, remplacez `endpoint` et exécutez la commande suivante.

```
kubectl get --raw endpoint
```

Cette commande vous permet de transmettre n'importe quel chemin de point de terminaison et renvoie la réponse brute. La sortie répertorie différentes métriques line-by-line, chaque ligne incluant un nom de métrique, des balises et une valeur.

```
metric_name{tag="value"[,...]} value
```

Récupérez les métriques depuis le serveur d'API

Le point de terminaison général du serveur d'API est exposé sur le plan de contrôle Amazon EKS. Ce point de terminaison est principalement utile lorsque vous examinez une métrique spécifique.

```
kubectl get --raw /metrics
```

L'exemple qui suit illustre un résultat.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

Cette sortie brute renvoie tel quel ce que le serveur d'API expose.

Récupérez les métriques du plan de contrôle avec `metrics.eks.amazonaws.com`

Pour les clusters de versions Kubernetes ou supérieures`1.28`, Amazon EKS expose également les métriques du groupe d'API. `metrics.eks.amazonaws.com` Ces mesures incluent les composants du plan de contrôle tels que `kube-scheduler` et`kube-controller-manager`.

> **Note**
>
> Si vous avez une configuration de webhook susceptible de bloquer la création de la nouvelle `APIService` ressource `v1.metrics.eks.amazonaws.com` sur votre cluster, il est possible que la fonctionnalité de point de terminaison des métriques ne soit pas disponible. Vous pouvez le vérifier dans le journal `kube-apiserver` d'audit en recherchant le `v1.metrics.eks.amazonaws.com` mot clé.

Récupérez **`kube-scheduler`** les métriques

Pour récupérer `kube-scheduler` des métriques, utilisez la commande suivante.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

L'exemple qui suit illustre un résultat.

```
# TYPE scheduler_pending_pods gauge
scheduler_pending_pods{queue="active"} 0
scheduler_pending_pods{queue="backoff"} 0
scheduler_pending_pods{queue="gated"} 0
scheduler_pending_pods{queue="unschedulable"} 18
# HELP scheduler_pod_scheduling_attempts [STABLE] Number of attempts to successfully
 schedule a pod.
# TYPE scheduler_pod_scheduling_attempts histogram
scheduler_pod_scheduling_attempts_bucket{le="1"} 79
scheduler_pod_scheduling_attempts_bucket{le="2"} 79
scheduler_pod_scheduling_attempts_bucket{le="4"} 79
scheduler_pod_scheduling_attempts_bucket{le="8"} 79
scheduler_pod_scheduling_attempts_bucket{le="16"} 79
scheduler_pod_scheduling_attempts_bucket{le="+Inf"} 81
[...]
```

Récupérez **`kube-controller-manager`** les métriques

Pour récupérer `kube-controller-manager` des métriques, utilisez la commande suivante.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

L'exemple qui suit illustre un résultat.

```
[...]
workqueue_work_duration_seconds_sum{name="pvprotection"} 0
workqueue_work_duration_seconds_count{name="pvprotection"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-08"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-07"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-05"} 19
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.001"} 109
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.01"} 139
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.1"} 181
workqueue_work_duration_seconds_bucket{name="replicaset",le="1"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="10"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="+Inf"} 191
workqueue_work_duration_seconds_sum{name="replicaset"} 4.265655885000002
[...]
```

Comprendre les métriques du planificateur et du responsable du contrôleur

Le tableau suivant décrit les métriques du planificateur et du gestionnaire de contrôleurs mises à disposition pour le scraping de style Prometheus. Pour plus d'informations sur ces métriques, consultez la référence des métriques Kubernetes dans la documentation Kubernetes.

Métrique	Composant du plan de contrôle	Description
scheduler_pending_pods	planificateur	Le nombre de pods qui attendent d'être planifiés sur un nœud pour être exécutés.
scheduler_schedule_attempts total	planificateur	Le nombre de tentatives effectuées pour planifier des pods.
planificateur_preemption_te ntats_total	planificateur	Le nombre de tentatives effectuées par le planificateur pour planifier des pods prioritai res en expulsant les pods moins prioritaires.

Métrique	Composant du plan de contrôle	Description
Scheduler_Preemption_Victims	planificateur	Le nombre de pods sélectionnés pour être expulsés afin de faire de la place à des pods prioritaires.
scheduler_pod_scheduling_attempts	planificateur	Le nombre de tentatives de planification réussie d'un Pod.
scheduler_scheduling_attempt duration_seconds	planificateur	Indique la rapidité ou la lenteur avec laquelle le planificateur est capable de trouver un emplacement approprié pour exécuter un Pod en fonction de divers facteurs tels que la disponibilité des ressources et les règles de planification.
scheduler_pod_scheduling_sli_duration_seconds	planificateur	La end-to-end latence d'un pod en cours de planification, à partir du moment où le pod entre dans la file d'attente de planification. Cela peut impliquer plusieurs tentatives de planification.
cronjob_controller_job_creation_skew_duration_seconds	responsable du contrôleur	Le délai entre le moment où l'exécution d'un cronjob est planifiée et le moment où le travail correspondant est créé.
profondeur de la file d'attente	responsable du contrôleur	Profondeur actuelle de la file d'attente.
workqueue_adds_total	responsable du contrôleur	Le nombre total d'ajouts gérés par la file de travail.

Métrique	Composant du plan de contrôle	Description
file d'attente de travail durée_duration_secondes	responsable du contrôleur	Durée en secondes pendant laquelle un élément reste dans la file d'attente avant d'être demandé.
workqueue_work_duration_seconds	responsable du contrôleur	Durée en secondes nécessaire au traitement d'un élément de la file d'attente.

Déployez un scraper Prometheus pour analyser régulièrement les indicateurs

Pour déployer un scraper Prometheus afin de récupérer régulièrement les métriques, utilisez la configuration suivante :

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-conf
data:
  prometheus.yml: |-
    global:
      scrape_interval: 30s
    scrape_configs:
    # apiserver metrics
    - job_name: apiserver-metrics
      kubernetes_sd_configs:
      - role: endpoints
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
```

```
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
    # Scheduler metrics
    - job_name: 'ksh-metrics'
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
    # Controller Manager metrics
    - job_name: 'kcm-metrics'
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
---
```

```
apiVersion: v1
kind: Pod
metadata:
  name: prom-pod
spec:
  containers:
  - name: prom-container
    image: prom/prometheus
    ports:
    - containerPort: 9090
    volumeMounts:
    - name: config-volume
      mountPath: /etc/prometheus/
  volumes:
  - name: config-volume
    configMap:
      name: prometheus-conf
```

L'autorisation qui suit est requise pour que le Pod puisse accéder au nouveau point de terminaison des métriques.

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

Pour corriger le rôle utilisé, vous pouvez utiliser la commande suivante.

```
kubectl patch clusterrole <role-name> --type=json -p='[
  {
    "op": "add",
    "path": "/rules/-",
    "value": {
      "verbs": ["get"],
      "apiGroups": ["metrics.eks.amazonaws.com"],
```

```
      "resources": ["kcm/metrics", "ksh/metrics"]
    }
  }
]'
```

Vous pouvez ensuite consulter le tableau de bord Prometheus en connectant le port du Prometheus scraper à votre port local.

```
kubectl port-forward pods/prom-pod 9090:9090
```

Pour votre cluster Amazon EKS, les principales métriques du plan de contrôle Kubernetes sont également ingérées dans CloudWatch Amazon Metrics sous l'espace de noms. `AWS/EKS` Pour les consulter, ouvrez la CloudWatch console et sélectionnez Toutes les mesures dans le volet de navigation de gauche. Sur la page de sélection des métriques, choisissez l'espace de `AWS/EKS` noms et une dimension de métriques pour votre cluster.

Surveillez les données du cluster avec Amazon CloudWatch

Amazon CloudWatch est un service de surveillance qui collecte des statistiques et des journaux à partir de vos ressources cloud. CloudWatch fournit gratuitement des métriques Amazon EKS de base lorsque vous utilisez un nouveau cluster de version `1.28` ou supérieure. Toutefois, lorsque vous utilisez l'opérateur CloudWatch d'observabilité en tant que module complémentaire Amazon EKS, vous pouvez bénéficier de fonctionnalités d'observabilité améliorées.

Statistiques de base sur Amazon CloudWatch

Pour les clusters de versions `1.28` Kubernetes ou supérieures, vous pouvez obtenir des métriques vendues gratuitement CloudWatch dans l'espace de noms. `AWS/EKS` Le tableau suivant fournit une liste des mesures de base disponibles pour les versions prises en charge. Chaque métrique répertoriée a une fréquence d'une minute.

Nom des métriques	Description
`scheduler_schedule_attempts_total`	Nombre total de tentatives effectuées par le planificateur pour planifier des pods dans le cluster pendant une période donnée. Cette métrique permet de surveiller la charge de travail du planificateur et peut indiquer la

Nom des métriques	Description
	pression de planification ou les problèmes potentiels liés au placement des pods. Unités : Nombre Statistiques valides : Sum
`scheduler_schedule_attempts_SCHEDULED`	Le nombre de tentatives réussies du planificateur pour programmer des pods vers des nœuds du cluster pendant une période donnée. Unités : Nombre Statistiques valides : Sum
`scheduler_schedule_attempts_UNSCHEDULABLE`	Le nombre de tentatives de planification de pods qui n'étaient pas planifiables pour une période donnée en raison de contraintes valides, telles que l'insuffisance du processeur ou de la mémoire sur un nœud. Unités : Nombre Statistiques valides : Sum
`scheduler_schedule_attempts_ERROR`	Le nombre de tentatives de planification de pods qui ont échoué pendant une période donnée en raison d'un problème interne lié au planificateur lui-même, tel que des problèmes de connectivité au serveur API. Unités : Nombre Statistiques valides : Sum

Nom des métriques	Description
`scheduler_pending_pods`	Le nombre total de pods en attente à planifier par le planificateur dans le cluster pour une période donnée. Unités : Nombre Statistiques valides : Sum
`scheduler_pending_pods_ACTIVEQ`	Le nombre de pods en attente dans ActiveQ, qui attendent d'être planifiés dans le cluster pour une période donnée. Unités : Nombre Statistiques valides : Sum
`scheduler_pending_pods_UNSCHEDULABLE`	Le nombre de pods en attente que le planificateur a tenté de planifier mais qui ont échoué, et qui sont maintenus dans un état non planifiable en cas de nouvelle tentative. Unités : Nombre Statistiques valides : Sum
`scheduler_pending_pods_BACKOFF`	Le nombre de pods en attente `backoffQ` dans un état d'attente qui attendent l'expiration de leur période d'attente. Unités : Nombre Statistiques valides : Sum

Nom des métriques	Description
`scheduler_pending_pods_GATED`	Le nombre de pods en attente qui attendent actuellement dans un état fermé, car ils ne peuvent pas être programmés tant qu'ils ne répondent pas aux conditions requises. Unités : Nombre Statistiques valides : Sum
`apiserver_request_total`	Le nombre de requêtes HTTP effectuées sur tous les serveurs d'API du cluster. Unités : Nombre Statistiques valides : Sum
`apiserver_request_total_4XX`	Nombre de requêtes HTTP adressées à tous les serveurs d'API du cluster qui ont généré des codes d'état `4XX` (erreur client). Unités : Nombre Statistiques valides : Sum
`apiserver_request_total_429`	Nombre de requêtes HTTP adressées à tous les serveurs d'API du cluster qui ont généré un code d'`429`état, ce qui se produit lorsque les clients dépassent les seuils de limitation de débit. Unités : Nombre Statistiques valides : Sum

Nom des métriques	Description
`apiserver_request_total_5XX`	Nombre de requêtes HTTP adressées à tous les serveurs d'API du cluster qui ont généré des codes d'état `5XX` (erreur de serveur). Unités : Nombre Statistiques valides : Sum
`apiserver_request_total_LIST_PODS`	Le nombre de requêtes `LIST` Pods adressées à tous les serveurs d'API du cluster. Unités : Nombre Statistiques valides : Sum
`apiserver_request_duration_seconds_PUT_P99`	99e percentile de latence pour les `PUT` demandes calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les `PUT` demandes sont traitées. Unités : secondes Statistiques valides : moyenne
`apiserver_request_duration_seconds_PATCH_P99`	99e percentile de latence pour les `PATCH` demandes calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les `PATCH` demandes sont traitées. Unités : secondes Statistiques valides : moyenne

Nom des métriques	Description
`apiserver_request_duration_seconds_POST_P99`	99e percentile de latence pour les `POST` demandes calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les `POST` demandes sont traitées. Unités : secondes Statistiques valides : moyenne
`apiserver_request_duration_seconds_GET_P99`	99e percentile de latence pour les `GET` demandes calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les `GET` demandes sont traitées. Unités : secondes Statistiques valides : moyenne
`apiserver_request_duration_seconds_LIST_P99`	99e percentile de latence pour les `LIST` demandes calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les `LIST` demandes sont traitées. Unités : secondes Statistiques valides : moyenne

Nom des métriques	Description
`apiserver_request_duration_seconds_DELETE_P99`	99e percentile de latence pour les `DELETE` demandes calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les `DELETE` demandes sont traitées. Unités : secondes Statistiques valides : moyenne
`apiserver_current_inflight_requests_MUTATING`	Le nombre de demandes mutantes (`POST`,, `PUTDELETE`,`PATCH`) en cours de traitement sur tous les serveurs d'API du cluster. Cette métrique représente les demandes en cours de traitement dont le traitement n'est pas encore terminé. Unités : Nombre Statistiques valides : Sum
`apiserver_current_inflight_requests_READONLY`	Le nombre de demandes en lecture seule (`GET`,`LIST`) actuellement traitées sur tous les serveurs d'API du cluster. Cette métrique représente les demandes en cours de traitement dont le traitement n'est pas encore terminé. Unités : Nombre Statistiques valides : Sum

Nom des métriques	Description
`apiserver_admission_webhook_request_total`	Le nombre de demandes de webhook d'admission effectuées sur tous les serveurs d'API du cluster. Unités : Nombre Statistiques valides : Sum
`apiserver_admission_webhook_request_total_ADMIT`	Le nombre de demandes de webhook d'admission mutantes effectuées sur tous les serveurs d'API du cluster. Unités : Nombre Statistiques valides : Sum
`apiserver_admission_webhook_request_total_VALIDATING`	Le nombre de demandes de webhook d'admission validantes effectuées sur tous les serveurs d'API du cluster. Unités : Nombre Statistiques valides : Sum
`apiserver_admission_webhook_rejection_count`	Le nombre de demandes d'admission au webhook effectuées sur tous les serveurs d'API du cluster qui ont été rejetées. Unités : Nombre Statistiques valides : Sum
`apiserver_admission_webhook_rejection_count_ADMIT`	Le nombre de demandes de webhook d'admission mutantes effectuées sur tous les serveurs d'API du cluster qui ont été rejetées. Unités : Nombre Statistiques valides : Sum

Nom des métriques	Description
`apiserver_admission_webhook_rejection_count_VALIDATING`	Le nombre de demandes de webhook d'admission en cours de validation effectuées sur tous les serveurs d'API du cluster qui ont été rejetées. Unités : Nombre Statistiques valides : Sum
`apiserver_admission_webhook_admission_duration_seconds`	Le 99e percentile de latence pour les demandes webhook d'admission tierces est calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les demandes de webhook d'admission de tiers sont traitées. Unités : secondes Statistiques valides : moyenne
`apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99`	Le 99e percentile de latence pour les demandes webhook d'admission mutantes tierces est calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les demandes de webhook d'admission mutantes tierces sont traitées. Unités : secondes Statistiques valides : moyenne

Nom des métriques	Description
`apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99`	99e percentile de latence pour les demandes webhook d'admission validées par des tiers, calculé à partir de toutes les demandes sur tous les serveurs d'API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les demandes de webhook d'admission validées par des tiers sont traitées. Unités : secondes Statistiques valides : moyenne
`apiserver_storage_size_bytes`	Taille physique en octets du fichier de base de données de stockage etcd utilisé par les serveurs d'API du cluster. Cette métrique représente l'espace disque réellement alloué au stockage. Unités : octets Statistiques valides : Maximum

Opérateur Amazon CloudWatch Observability

Amazon CloudWatch Observability collecte des journaux, des métriques et des données de suivi en temps réel. Il les envoie à Amazon CloudWatch et à AWS X-Ray. Vous pouvez installer ce module complémentaire pour activer à la fois CloudWatch Application Signals et CloudWatch Container Insights avec une observabilité améliorée pour Amazon EKS. Cela vous aide à surveiller l'état et les performances de votre infrastructure et de vos applications conteneurisées. L'opérateur Amazon CloudWatch Observability est conçu pour installer et configurer les composants nécessaires.

Amazon EKS prend en charge l'opérateur CloudWatch d'observabilité en tant que module complémentaire Amazon EKS. Le module complémentaire permet d'utiliser Container Insights sur les nœuds de travail Linux et Windows du cluster. Pour activer Container Insights sous Windows, la version du module complémentaire Amazon EKS doit être `1.5.0` ou supérieure. Actuellement, CloudWatch Application Signals n'est pas pris en charge sur Amazon EKS Windows.

Les rubriques ci-dessous décrivent comment commencer à utiliser CloudWatch Observability Operator pour votre cluster Amazon EKS.

- Pour obtenir des instructions sur l'installation de ce module complémentaire, consultez Installer l' CloudWatch agent avec le module complémentaire Amazon CloudWatch Observability EKS ou le tableau Helm dans le guide de l' CloudWatch utilisateur Amazon.
- Pour plus d'informations sur les signaux CloudWatch d'application, consultez la section Signaux d'application dans le guide de CloudWatch l'utilisateur Amazon.
- Pour plus d'informations sur Container Insights, consultez la section Utilisation de Container Insights dans le guide de CloudWatch l'utilisateur Amazon.

Envoyer les journaux du plan de contrôle à CloudWatch Logs

La journalisation du plan de contrôle Amazon EKS fournit des journaux d'audit et de diagnostic directement depuis le plan de contrôle Amazon EKS vers CloudWatch les journaux de votre compte. Ces journaux vous permettent de sécuriser et d'exécuter facilement vos clusters. Vous pouvez sélectionner les types de journaux exacts dont vous avez besoin, et les journaux sont envoyés sous forme de flux de journaux à un groupe pour chaque cluster Amazon EKS inclus CloudWatch. Vous pouvez utiliser des filtres CloudWatch d'abonnement pour analyser les journaux en temps réel ou pour les transmettre à d'autres services (les journaux seront codés en Base64 et compressés au format gzip). Pour plus d'informations, consultez Amazon CloudWatch logging.

Vous pouvez commencer à utiliser la journalisation de plan de contrôle d'Amazon EKS en choisissant les types de journal que vous voulez activer pour chaque cluster Amazon EKS nouveau ou existant. Vous pouvez activer ou désactiver chaque type de journal par cluster à l'aide de la AWS Management Console AWS CLI (version `1.16.139` ou supérieure) ou via l'API Amazon EKS. Lorsque cette option est activée, les journaux sont automatiquement envoyés depuis le cluster Amazon EKS aux CloudWatch journaux du même compte.

Lorsque vous utilisez la journalisation sur le plan de contrôle Amazon EKS, le tarif Amazon EKS standard vous est facturé pour chaque cluster que vous exécutez. Les frais standard d'ingestion et de stockage des données CloudWatch Logs vous sont facturés pour tous les journaux envoyés à CloudWatch Logs depuis vos clusters. Vous êtes également facturé pour toutes les AWS ressources, telles que les EC2 instances Amazon ou les volumes Amazon EBS, que vous fournissez dans le cadre de votre cluster.

Les types de journal de plan de contrôle de cluster suivants sont disponibles. Chaque type de journal correspond à un composant du plan de contrôle Kubernetes. Pour en savoir plus sur ces composants, consultez Kubernetes Components dans la documentation Kubernetes.

Serveur d'API**`api`** ()

Le serveur d'API de votre cluster est le composant du plan de contrôle qui expose l'API Kubernetes. Si vous activez les journaux du serveur d'API au moment du lancement du cluster, ou peu après, les journaux incluent les indicateurs qui ont été utilisés pour démarrer le serveur d'API. Pour plus d'informations, consultez kube-apiserver et la stratégie d'audit dans la documentation Kubernetes.

Audit**`audit`** ()

Les journaux d'audit Kubernetes fournissent un enregistrement des utilisateurs individuels, des administrateurs ou des composants du système qui ont affecté votre cluster. Pour plus d'informations, consultez la section Audit dans la documentation Kubernetes.

Authentificateur**`authenticator`** ()

Les journaux d'authentification sont propres à Amazon EKS. Ces journaux représentent le composant de plan de contrôle utilisé par Amazon EKS pour l'authentification de contrôle d'accès basé sur les rôles (RBAC) Kubernetes à l'aide d'informations d'identification IAM. Pour de plus amples informations, veuillez consulter Gestion du cluster.

Gestionnaire de contrôleur**`controllerManager`** ()

Le gestionnaire de contrôleurs gère les boucles de contrôle principales fournies avec Kubernetes. Pour plus d'informations, consultez kube-controller-manager dans la documentation Kubernetes.

Planificateur**`scheduler`** ()

Le composant planificateur gère quand et où exécuter les pods dans votre cluster. Pour plus d'informations, consultez kube-scheduler dans la documentation Kubernetes.

Activer ou désactiver les journaux du plan de contrôle

Par défaut, les journaux du plan de contrôle du cluster ne sont pas envoyés à CloudWatch Logs. Vous devez activer chaque type de journal individuellement pour envoyer des journaux pour votre cluster. CloudWatch Les taux d'ingestion des journaux, de stockage des archives et d'analyse des données s'appliquent aux journaux du plan de contrôle activés. Pour en savoir plus, consultez PricingCloudWatch (Tarification).

Pour mettre à jour la configuration de la journalisation du plan de contrôle, Amazon EKS a besoin d'un maximum de cinq adresses IP disponibles dans chaque sous-réseau. Lorsque vous activez un type de journal, les journaux sont envoyés avec un niveau de détail de journal 2.

Vous pouvez activer ou désactiver les journaux du plan de contrôle à l'aide de la CLI AWS Management Consoleou de la AWS CLI.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Sélectionnez l'onglet Observabilité.
4. Dans la section Journalisation du plan de contrôle, sélectionnez Gérer la journalisation.
5. Pour chaque type de journal, choisissez si celui-ci doit être activé ou désactivé. Par défaut, chaque type de journal est désactivé.
6. Choisissez Save changes (Enregistrer les modifications) pour terminer.

AWS CLI

1. Vérifiez la version de votre AWS CLI à l'aide de la commande suivante.

   ```
   aws --version
   ```

   Si la version de votre AWS CLI est antérieure à`1.16.139`, vous devez d'abord effectuer la mise à jour vers la dernière version. Pour installer ou mettre à niveau la AWS CLI, reportez-vous à la section Installation de l'interface de ligne de AWS commande dans le guide de l'utilisateur de l'interface de ligne de AWS commande.

2. Mettez à jour la configuration d'exportation du journal du plan de contrôle de votre cluster à l'aide de la commande AWS CLI suivante. Remplacez *`my-cluster`* par le nom de votre cluster et spécifiez les valeurs d'accès au point de terminaison souhaitées.

   > **Note**
   >
   > La commande suivante envoie tous les types de journaux disponibles à CloudWatch Logs.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. Surveillez l'état de la mise à jour de votre configuration de journal avec la commande suivante, en indiquant le nom de votre cluster et l'ID de mise à jour qui ont été renvoyés par la commande précédente. Votre mise à jour est terminée lorsqu'elle affiche l'état `Successful`.

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
```

```
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

Afficher les journaux du plan de contrôle du cluster

Une fois que vous avez activé l'un des types de journaux du plan de contrôle pour votre cluster Amazon EKS, vous pouvez les consulter sur la CloudWatch console.

Pour en savoir plus sur l'affichage, l'analyse et la gestion des connexions CloudWatch, consultez le guide de l'utilisateur Amazon CloudWatch Logs.

1. Ouvrez la CloudWatch console. Ce lien ouvre la console et affiche vos groupes de journaux disponibles actuels et les filtre avec le préfixe `/aws/eks`.
2. Choisissez le cluster pour lequel vous souhaitez afficher les journaux. Le format du nom de groupe de journaux est `/aws/eks/`*`my-cluster`*`/cluster`.
3. Choisissez le flux de journal à afficher. La liste suivante décrit le format de nom de flux du journal pour chaque type de journal.

> **Note**
>
> À mesure que les données de flux de journal augmentent, les noms de flux de journal font l'objet d'une rotation. Lorsqu'il existe plusieurs flux de journaux pour un type de journal particulier, vous pouvez afficher le dernier flux en recherchant son nom avec l'heure du dernier événement (Last event time).

- Journaux de composant de serveur d'API Kubernetes (**`api`**) : `kube-apiserver-`*`1234567890abcdef01234567890abcde`*

- Audit (**audit** : `kube-apiserver-audit-`*`1234567890abcdef01234567890abcde`*
- Authentificateur (**authenticator**) : `authenticator-`*`1234567890abcdef01234567890abcde`*
- Gestionnaire de contrôleur (**controllerManager**) : `kube-controller-manager-`*`1234567890abcdef01234567890abcde`*
- Planificateur (**scheduler**) : `kube-scheduler-`*`1234567890abcdef01234567890abcde`*

4. Parcourez les événements du flux de journaux.

   Par exemple, vous devez voir les indicateurs de serveur d'API initiaux correspondant au cluster lorsque vous consultez le haut de `kube-apiserver-`*`1234567890abcdef01234567890abcde`*
   .

   > **Note**
   >
   > Si les journaux du serveur d'API ne s'affichent pas au début du flux de journaux, il est probable que le fichier journal du serveur d'API ait fait l'objet d'une rotation sur le serveur avant que vous n'activiez la journalisation du serveur d'API sur le serveur. Les fichiers journaux qui font l'objet d'une rotation avant que la journalisation du serveur d'API ne soit activée ne peuvent pas être exportés vers CloudWatch.

Toutefois, vous pouvez créer un nouveau cluster avec la même version de Kubernetes et activer la journalisation du serveur API lorsque vous créez le cluster. Les clusters dotés de la même version de plate-forme ont les mêmes drapeaux activés. Vos drapeaux doivent donc correspondre aux drapeaux du nouveau cluster. Lorsque vous avez fini de visualiser les drapeaux du nouveau cluster dans CloudWatch, vous pouvez supprimer le nouveau cluster.

Consigner les appels d'API en tant qu' AWS CloudTrail événements

Amazon EKS est intégré à AWS CloudTrail. CloudTrail est un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un AWS service dans Amazon EKS. CloudTrail capture tous les appels d'API pour Amazon EKS sous forme d'événements. Cela inclut les appels de la console Amazon EKS et les appels de code aux opérations de l'API Amazon EKS.

Si vous créez un suivi, vous pouvez activer la diffusion continue des CloudTrail événements vers un compartiment Amazon S3. Cela inclut les événements pour Amazon EKS. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail

console dans Historique des événements. À l'aide des informations CloudTrail collectées, vous pouvez déterminer plusieurs détails concernant une demande. Par exemple, vous pouvez déterminer quand la demande a été envoyée à Amazon EKS, l'adresse IP à partir de laquelle la demande a été effectuée et qui a effectué la demande.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Rubriques

- Consultez des références utiles pour AWS CloudTrail
- Analyser les entrées du fichier AWS CloudTrail journal
- Afficher les statistiques des groupes Amazon EC2 Auto Scaling

Consultez des références utiles pour AWS CloudTrail

Lorsque vous créez votre AWS compte, elle CloudTrail est également activée sur votre AWS compte. Chaque fois qu'une activité se produit dans Amazon EKS, elle est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements pour Amazon EKS, créez un historique. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un parcours dans la console, celui-ci s'applique à toutes les AWS régions. Le journal enregistre les événements de toutes les AWS régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, veuillez consulter les ressources suivantes.

- Présentation de la création d'un journal de suivi
- CloudTrail services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions Amazon EKS sont enregistrées CloudTrail et documentées dans le manuel Amazon EKS API Reference. Par exemple, les appels aux DeleteClustersections CreateCluster, ListClusterset génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des détails sur le type d'identité IAM ayant effectué la demande, ainsi que les informations d'identification qui ont été utilisées. Si des informations d'identification temporaires ont été utilisées, l'entrée montre comment ces informations d'identification ont été obtenues.

Pour de plus amples informations, veuillez consulter l'élément userIdentity CloudTrail .

Analyser les entrées du fichier AWS CloudTrail journal

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande émise par une source et comprend des informations sur l'action demandée. Cela inclut des informations telles que la date et l'heure de l'action et les paramètres de demande utilisés. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`CreateCluster`action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws: iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
```

```
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws: eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws: iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Entrées de journal des rôles liés à un service Amazon EKS

Les rôles liés au service Amazon EKS effectuent des appels d'API vers AWS des ressources. CloudTrail enregistre les entrées avec `username: AWSServiceRoleForAmazonEKS` et `username: AWSServiceRoleForAmazonEKSNodegroup` apparaît pour les appels effectués par le service Amazon EKS avec des rôles liés. Pour plus d'informations sur Amazon EKS et les rôles liés aux services, consultez the section called “Rôles liés à un service”.

L'exemple suivant montre une entrée de CloudTrail journal illustrant une `DeleteInstanceProfile`action effectuée par le rôle lié au `AWSServiceRoleForAmazonEKSNodegroup` service, indiqué dans le`sessionContext`.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws: sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws: iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
```

```
    "recipientAccountId": "111122223333"
}
```

Afficher les statistiques des groupes Amazon EC2 Auto Scaling

Les métriques de groupe Amazon EC2 Auto Scaling sont activées par défaut pour les groupes de nœuds gérés par Amazon EKS, sans frais supplémentaires. Le groupe Auto Scaling envoie des échantillons de données à Amazon CloudWatch toutes les minutes. Ces mesures peuvent être affinées par le nom des groupes Auto Scaling. Ils vous offrent une visibilité continue sur l'historique du groupe Auto Scaling qui alimente vos groupes de nœuds gérés, notamment sur l'évolution de la taille du groupe au fil du temps. Les métriques de groupe Auto Scaling sont disponibles dans la console Amazon CloudWatch ou Auto Scaling. Pour plus d'informations, consultez la section Surveiller CloudWatch les métriques de vos groupes et instances Auto Scaling.

Grâce à la collecte des métriques de groupe Auto Scaling, vous pouvez surveiller le dimensionnement des groupes de nœuds gérés. Les métriques de groupe Auto Scaling indiquent les tailles minimale, maximale et souhaitée d'un groupe Auto Scaling. Vous pouvez créer une alarme si le nombre de nœuds d'un groupe de nœuds est inférieur à la taille minimale, ce qui indique que le groupe est défectueux. Le suivi de la taille des groupes de nœuds est également utile pour ajuster le nombre maximum afin que votre plan de données ne soit pas à court de capacité.

Si vous préférez que ces statistiques ne soient pas collectées, vous pouvez choisir de les désactiver toutes ou seulement certaines d'entre elles. Vous pouvez par exemple le faire pour éviter le bruit dans vos CloudWatch tableaux de bord. Pour plus d'informations, consultez Amazon CloudWatch Metrics pour Amazon EC2 Auto Scaling.

Envoyer des données métriques et de suivi avec ADOT Operator

Amazon EKS prend en charge l' AWS Management Console utilisation de la AWS CLI et de l'API Amazon EKS pour installer et gérer l'AWS opérateur Distro for OpenTelemetry (ADOT). Cela permet à vos applications exécutées sur Amazon EKS d'envoyer plus facilement des données métriques et de suivi à plusieurs options de service de surveillance telles qu'Amazon CloudWatch, Prometheus et X-Ray.

Pour plus d'informations, consultez Getting Started with AWS Distro pour OpenTelemetry l'utilisation des modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

Afficher les données agrégées sur les ressources du cluster avec le tableau de bord EKS

image::images/eks-dashboard.png[screenshot of account level cluster metrics]

Qu'est-ce que le tableau de bord Amazon EKS ?

Le tableau de bord Amazon EKS fournit une visibilité consolidée sur vos clusters Kubernetes dans plusieurs AWS régions et comptes. AWS Grâce à ce tableau de bord, vous pouvez :

- Suivez les clusters dont les end-of-support mises à niveau automatiques sont prévues dans les 90 prochains jours.
- Coûts du plan de contrôle du projet EKS pour les clusters bénéficiant d'un support étendu.
- Passez en revue les clusters avec des informations qui nécessitent une attention particulière avant la mise à niveau.
- Identifiez les groupes de nœuds gérés exécutant des versions d'AMI spécifiques.
- Surveillez la distribution des types de support du cluster (standard par rapport au support étendu).

Le tableau de bord EKS s'intègre à EKS Cluster Insights pour détecter les problèmes liés à vos clusters, tels que l'utilisation de Kubernetes obsolètes. APIs Pour de plus amples informations, veuillez consulter the section called “Informations sur le cluster”.

> **Note**
>
> Le tableau de bord EKS n'est pas en temps réel et est mis à jour toutes les 12 heures. Pour la surveillance des clusters en temps réel, voir Surveiller les clusters

Comment le tableau de bord utilise-t-il AWS les Organizations ?

Le tableau de bord Amazon EKS nécessite l'intégration AWS des Organizations pour fonctionner. Il permet aux AWS Organisations de recueillir en toute sécurité des informations sur les clusters entre les comptes. Cette intégration permet une gestion et une gouvernance centralisées à mesure que votre AWS infrastructure évolue.

Si l'option AWS Organizations n'est pas activée pour votre infrastructure, consultez le Guide de l'utilisateur AWS des organisations pour obtenir des instructions de configuration.

Accès entre régions et entre comptes

Le tableau de bord EKS peut voir les ressources du cluster dans n'importe quel compte membre de l' AWS organisation. Pour générer une liste des AWS comptes de votre organisation, voir Exporter les détails de tous les comptes d'une organisation.

La région AWS us-east-1 génère le tableau de bord. Vous devez vous connecter à cette région pour voir le tableau de bord. Le tableau de bord agrège les données entre AWS les régions, mais cela n'inclut pas les `GovCloud` régions de Chine.

Termes clés

- AWS Organisation : structure de gestion unifiée pour plusieurs AWS comptes.
- Compte de gestion : compte principal qui contrôle l' AWS organisation.
- Compte membre : n'importe quel compte au sein de l'organisation, à l'exception du compte de gestion.
- Administrateur délégué : un compte membre a accordé des autorisations administratives spécifiques entre comptes. Dans le compte de gestion, vous pouvez sélectionner un compte d'administrateur délégué par AWS service.
- Accès sécurisé : autorisation pour le tableau de bord EKS d'accéder aux informations du cluster via les comptes de l'organisation.
- Rôle lié à un service (SLRs) : type unique de rôle IAM directement lié à un service. AWS Le tableau de bord EKS utilise un appareil reflex pour lire les informations relatives à vos comptes et à vos organisations.
- Pour plus d'informations, reportez-vous à la section Terminologie et concepts du Guide de l'utilisateur AWS des Organizations.

Vue d'ensemble générale

1. Accédez au compte de gestion de votre AWS organisation.
   - Les étapes pour accéder au compte de gestion dépendent de la manière dont vous avez configuré votre AWS organisation. Par exemple, vous pouvez accéder au compte de gestion via AWS Identity Center ou Okta.
2. Activez l'accès sécurisé via la console EKS.
3. Désignez un administrateur délégué à l'aide de son identifiant de AWS compte.

4. Passez au compte d'administrateur délégué.
5. Accédez à la console EKS améliorée avec une visibilité à l'échelle de l'entreprise.

Activez le tableau de bord EKS à l'aide de la AWS console

> **Important**
>
> Vous devez être connecté au compte de gestion de votre AWS organisation pour activer le tableau de bord EKS.

Accédez aux paramètres du tableau de bord EKS

1. Assurez-vous des points suivants :
   a. AWS Organizations est activée et configurée.
   b. Vous êtes connecté au compte de gestion de l'organisation.
   c. Vous consultez la console de AWS gestion dans la région us-east-1.
2. Accédez à la console EKS.
3. Dans la barre latérale gauche, ouvrez les paramètres du tableau de bord.

Configurer l'accès au tableau de bord Amazon EKS

1. Trouvez l'identifiant du AWS AWS compte que vous souhaitez autoriser à consulter le tableau de bord EKS.
   a. Cette étape est facultative, mais suggérée. Si ce n'est pas le cas, vous ne pouvez accéder au tableau de bord qu'à partir du compte de gestion. Il est recommandé de limiter l'accès au compte de gestion.
2. Cliquez sur Activer l'accès sécurisé.
   a. Vous pouvez désormais consulter le tableau de bord depuis le compte de gestion.
3. Cliquez sur Enregistrer un administrateur délégué et saisissez l'identifiant du AWS compte que vous utiliserez pour consulter le tableau de bord.
   a. Vous pouvez désormais consulter le tableau de bord depuis le compte d'administrateur délégué ou depuis le compte de gestion.

Pour plus d'informations sur les autorisations requises pour activer le tableau de bord, consultez la section Politiques IAM minimales requises.

Afficher le tableau de bord EKS

1. Connectez-vous au compte d'administrateur délégué (suggéré) ou au compte de gestion.
2. Connectez-vous à la région us-east-1.
3. Accédez au service EKS, puis sélectionnez Tableau de bord dans la barre latérale gauche.

> **Note**
>
> Vérifiez les autorisations IAM requises pour consulter le tableau de bord EKS.

Configuration du tableau de bord

Vous pouvez configurer l'affichage du tableau de bord et filtrer les ressources.

Ressources disponibles

- Clusters : affichez des informations agrégées sur l'état et l'emplacement des clusters EKS.
  - Clusters présentant des problèmes de santé.
  - Clusters sur EKS Extended Support.
  - Répartition des clusters par version de Kubernetes.
- Groupes de nœuds gérés : passez en revue les groupes de nœuds gérés et EC2 les instances.
  - Groupes de nœuds par type d'AMI, tels qu'Amazon Linux ou Bottlerocket.
  - Problèmes de santé des groupes de nœuds.
  - Distribution des types d'instances.
- Modules complémentaires : découvrez les modules complémentaires Amazon EKS que vous avez installés et leur statut.
  - Nombre d'installations par module complémentaire.
  - Modules complémentaires présentant des problèmes de santé.
  - Distribution des versions par module complémentaire.

Vues disponibles

- Vue graphique
  - Une vue de widget personnalisable affichant des graphiques et des visualisations de la ressource sélectionnée.
  - Les modifications apportées à la vue graphique, telles que la suppression d'un widget, sont visibles par tous les utilisateurs du tableau de bord EKS.
- Vue des ressources
  - Vue sous forme de liste de la ressource sélectionnée, filtres compatibles.
- Vue de la carte
  - Affichez la distribution géographique de la ressource sélectionnée.

Filtrer le tableau de bord EKS

Vous pouvez filtrer le tableau de bord EKS comme suit :

- AWS Compte
- Unité organisationnelle, définie par AWS Organizations
- AWS Région

Désactiver le tableau de bord EKS à l'aide de la AWS console

1. Assurez-vous des points suivants :
   a. AWS Organizations est activée et configurée.
   b. Vous êtes connecté au compte de gestion de l'organisation.
   c. Vous consultez la console de AWS gestion dans la région us-east-1.
2. Accédez à la console EKS.
3. Dans la barre latérale gauche, ouvrez les paramètres du tableau de bord.
4. Cliquez sur Désactiver l'accès sécurisé.

Résoudre les problèmes liés au tableau de bord EKS

Problème d'activation du tableau de bord EKS

- Vous devez être connecté au compte de gestion d'une AWS organisation.
  - Si vous n'avez pas d' AWS organisation, créez-en une. Découvrez comment créer et configurer une organisation.
  - Si votre AWS compte est déjà membre d'une AWS organisation, identifiez l'administrateur de l'organisation.
- Vous devez être connecté au AWS compte avec des autorisations IAM suffisantes pour créer et mettre à jour les ressources AWS des Organizations.

Problème lors de l'affichage du tableau de bord EKS

- Vous devez être connecté à l'un des AWS comptes suivants :
  - Le compte de gestion de l' AWS Organisation
  - Un compte d'administrateur délégué, identifié dans les paramètres du tableau de bord EKS du compte de gestion.
- Si vous avez récemment activé le tableau de bord EKS, veuillez noter que la saisie initiale des données peut prendre jusqu'à 12 heures.
- Essayez de réactiver le tableau de bord à l'aide de la CLI, notamment en créant le rôle lié au service.

Les widgets du tableau de bord se déplacent de

- Le tableau de bord EKS enregistre la vue du widget configurable au niveau du AWS compte. Si vous modifiez l'affichage du widget, les autres utilisateurs du même AWS compte verront les modifications.

Configurer l'intégration du tableau de bord EKS avec les AWS Organizations

Cette section fournit des step-by-step instructions pour configurer l'intégration du tableau de bord EKS avec AWS Organizations. Vous apprendrez comment activer et désactiver l'accès sécurisé entre

les services, ainsi que comment enregistrer et désenregistrer des comptes d'administrateur délégué. Chaque tâche de configuration peut être exécutée à l'aide de la AWS console ou de la AWS CLI.

Activer l'accès approuvé

L'accès sécurisé autorise le tableau de bord EKS à accéder en toute sécurité aux informations du cluster sur tous les comptes de votre organisation.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Cliquez sur Activer l'accès sécurisé.

> **Note**
>
> Lorsque vous activez l'accès sécurisé via la console EKS, le système crée automatiquement le rôle `AWSServiceRoleForAmazonEKSDashboard` lié au service. Cette création automatique ne se produit pas si vous activez l'accès sécurisé à l'aide de la AWS CLI ou de la console AWS Organizations.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez les commandes suivantes :

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

Désactiver l'accès approuvé

La désactivation de l'accès sécurisé révoque l'autorisation du tableau de bord EKS d'accéder aux informations du cluster sur les comptes de votre organisation.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Cliquez sur Désactiver l'accès sécurisé.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez la commande suivante :

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

Activer un compte administrateur délégué

Un administrateur délégué est un compte membre autorisé à accéder au tableau de bord EKS.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Cliquez sur Enregistrer un administrateur délégué.
5. Entrez l'ID de compte du AWS compte que vous souhaitez choisir comme administrateur délégué.
6. Confirmez l'enregistrement.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez la commande suivante en `123456789012` remplaçant par votre identifiant de compte :

```
aws organizations register-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Désactiver un compte d'administrateur délégué

La désactivation d'un administrateur délégué supprime l'autorisation du compte d'accéder au tableau de bord EKS.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Localisez l'administrateur délégué dans la liste.
5. Cliquez sur Désenregistrer à côté du compte que vous souhaitez supprimer en tant qu'administrateur délégué.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez la commande suivante, en `123456789012` remplaçant par l'ID de compte de l'administrateur délégué :

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Politiques IAM minimales requises

Cette section décrit les politiques IAM minimales requises pour permettre un accès sécurisé et déléguer un administrateur pour l'intégration du tableau de bord EKS avec les AWS Organizations.

Politique visant à permettre un accès sécurisé

Pour activer un accès sécurisé entre EKS Dashboard et AWS Organizations, vous devez disposer des autorisations suivantes :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws: iam::*:role/aws-service-role/
dashboard.eks.amazonaws.com/AWSServiceRoleForAmazonEKSDashboard"
        }
    ]
}
```

Politique de délégation d'un administrateur

Pour enregistrer ou désenregistrer un administrateur délégué pour le tableau de bord EKS, vous devez disposer des autorisations suivantes :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique de consultation du tableau de bord EKS

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AmazonEKSDashboardReadOnly",
            "Effect": "Allow",
            "Action": [
                "eks:ListDashboardData",
                "eks:ListDashboardResources",
                "eks:DescribeClusterVersions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsReadOnly",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListRoots",
                "organizations:ListAccountsForParent",
                "organizations:ListOrganizationalUnitsForParent"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsDelegatedAdmin",
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "organizations:ServicePrincipal": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> Ces politiques doivent être associées au principal IAM (utilisateur ou rôle) dans le compte de gestion de votre AWS organisation. Les comptes membres ne peuvent pas permettre un accès sécurisé ou déléguer des administrateurs.

Améliorez EKS avec des AWS services intégrés

Outre les services décrits dans d'autres sections, Amazon EKS propose d'autres AWS services pour fournir des solutions supplémentaires. Cette rubrique identifie certains des autres services qui utilisent Amazon EKS pour ajouter des fonctionnalités, ou des services qu'Amazon EKS utilise pour effectuer des tâches.

Rubriques

- Créez des ressources Amazon EKS avec AWS CloudFormation
- Analysez les événements de sécurité sur EKS avec Amazon Detective
- Détectez les menaces avec Amazon GuardDuty
- Évaluez la résilience du cluster EKS avec AWS Resilience Hub
- Centralisez et analysez les données de sécurité EKS avec Security Lake
- Activez une connectivité sécurisée entre clusters avec Amazon VPC Lattice
- Lancez des clusters EKS à faible latence avec des Zones AWS Locales

Créez des ressources Amazon EKS avec AWS CloudFormation

Amazon EKS est intégré à AWS CloudFormation un service qui vous aide à modéliser et à configurer vos AWS ressources afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. Vous créez un modèle qui décrit toutes les AWS ressources que vous souhaitez, par exemple un cluster Amazon EKS, et vous vous AWS CloudFormation occupez du provisionnement et de la configuration de ces ressources pour vous.

Lorsque vous l'utilisez AWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources Amazon EKS de manière cohérente et répétée. Décrivez simplement vos ressources une seule fois, puis fournissez les mêmes ressources à plusieurs reprises dans plusieurs AWS comptes et régions.

Amazon EKS et AWS CloudFormation modèles

Pour allouer et configurer des ressources pour Amazon EKS et les services associés, vous devez comprendre les modèles AWS CloudFormation . Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez mettre à disposition dans

vos AWS CloudFormation piles. Si vous n'êtes pas familiarisé avec JSON ou YAML, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec les AWS CloudFormation modèles. Pour plus d'informations, voir Qu'est-ce que AWS CloudFormation Designer ? dans le guide de AWS CloudFormation l'utilisateur.

Amazon EKS prend en charge la création de clusters et de groupes de nœuds dans AWS CloudFormation. Pour plus d'informations, notamment des exemples de modèles JSON et YAML pour vos ressources Amazon EKS, consultez la référence au type de ressource Amazon EKS dans le guide de l' AWS CloudFormation utilisateur.

En savoir plus sur AWS CloudFormation

Pour en savoir plus AWS CloudFormation, consultez les ressources suivantes :

- AWS CloudFormation
- AWS CloudFormation Guide de l'utilisateur
- AWS CloudFormation Guide de l'utilisateur de l'interface de ligne de commande

Analysez les événements de sécurité sur EKS avec Amazon Detective

Amazon Detective vous permet d'analyser, d'enquêter et d'identifier rapidement la cause racine des résultats de sécurité ou des activités suspectes. Detective collecte automatiquement les données du journal à partir de vos AWS ressources. Detective utilise ensuite le machine learning, l'analyse statistique et la théorie des graphes pour générer des visualisations qui vous aideront à mener des investigations de sécurité plus rapides et plus efficaces. Les agrégations de données, les résumés et le contexte prédéfinis de Detective vous aident à analyser et à déterminer rapidement la nature et l'étendue des éventuels problèmes de sécurité. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon Detective.

Detective organise Kubernetes et les AWS données en résultats tels que :

- Les détails du cluster Amazon EKS, y compris l'identité IAM qui a créé le cluster et la fonction du service du cluster. Vous pouvez étudier l'activité de ces identités IAM AWS et l'activité de l'API Kubernetes avec Detective.
- Les détails du conteneur, tels que l'image et le contexte de sécurité. Vous pouvez également consulter les informations relatives aux pods résiliés.

- Activité des API Kubernetes, y compris les tendances générales de l'activité des API et les détails sur des appels d'API spécifiques. Par exemple, vous pouvez afficher le nombre d'appels d'API Kubernetes réussis et échoués qui ont été émis pendant une période sélectionnée. En outre, la section sur les appels d'API récemment observés peut être utile pour identifier une activité suspecte.

Les journaux d'audit Amazon EKS sont un package de source de données facultatif qui peut être ajouté à votre graphe de comportement Detective. Vous pouvez afficher les packages source facultatifs disponibles et leur état dans votre compte. Pour plus d'informations, veuillez consulter la rubrique Journaux d'audit d'Amazon EKS pour Detective dans le Guide de l'utilisateur Amazon Detective.

Utilisation d'Amazon Detective avec Amazon EKS

Avant de pouvoir examiner les résultats, Detective doit être activé pendant au moins 48 heures dans la même AWS région que celle dans laquelle se trouve votre cluster. Pour plus d'informations, veuillez consulter la rubrique Configuration d'Amazon Detective dans le Guide de l'utilisateur Amazon Detective.

1. Ouvrez la console Detective à l'adresse https://console.aws.amazon.com/detective/.
2. Dans le volet de navigation de gauche, sélectionnez Search (Rechercher).
3. Sélectionnez Choose type (Choisir le type), puis sélectionnez EKS cluster.
4. Saisissez le nom du cluster ou l'ARN, puis choisissez Search (Rechercher).
5. Dans les résultats de la recherche, sélectionnez le nom du cluster dont vous souhaitez consulter l'activité. Pour plus d'informations sur ce que vous pouvez consulter, consultez la section Activité globale de l'API Kubernetes impliquant un cluster Amazon EKS dans le manuel Amazon Detective User Guide.

Détectez les menaces avec Amazon GuardDuty

Amazon GuardDuty est un service de détection des menaces qui vous aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre AWS environnement. À l'aide de modèles d'apprentissage automatique (ML) et de fonctionnalités de détection des anomalies et des menaces, vous surveillez GuardDuty en permanence les différentes sources de journaux et l'activité d'exécution afin d'identifier et de hiérarchiser les risques de sécurité potentiels et les activités malveillantes dans votre environnement.

Parmi les autres fonctionnalités, GuardDuty propose les deux fonctionnalités suivantes qui détectent les menaces potentielles pour vos clusters EKS : la protection EKS et la surveillance du temps d'exécution.

> **Note**
>
> Nouveau : le mode automatique d'Amazon EKS s'intègre à GuardDuty.

Protection EKS

Cette fonctionnalité fournit une couverture de détection des menaces pour vous aider à protéger les clusters Amazon EKS en surveillant les journaux d'audit Kubernetes associés. Les journaux d'audit Kubernetes capturent les actions séquentielles au sein de votre cluster, notamment les activités des utilisateurs, des applications utilisant l'API Kubernetes et du plan de contrôle. Par exemple, GuardDuty peut identifier que les appels visant à altérer potentiellement les ressources d'un cluster Kubernetes ont été APIs appelés par un utilisateur non authentifié.

Lorsque vous activez EKS Protection, vous ne GuardDuty pourrez accéder à vos journaux d'audit Amazon EKS que pour une détection continue des menaces. S'il GuardDuty identifie une menace potentielle pour votre cluster, il génère une découverte associée au journal d'audit Kubernetes d'un type spécifique. Pour plus d'informations sur les types de résultats disponibles dans les journaux d'audit Kubernetes, consultez la section Types de résultats des journaux d'audit Kubernetes dans le guide de l'utilisateur Amazon. GuardDuty

Pour plus d'informations, consultez EKS Protection dans le guide de GuardDuty l'utilisateur Amazon.

Surveillance du temps d'exécution

Cette fonctionnalité surveille et analyse les événements au niveau du système d'exploitation, du réseau et des fichiers afin de vous aider à détecter les menaces potentielles liées à des AWS charges de travail spécifiques de votre environnement.

Lorsque vous activez la surveillance du temps d'exécution et que vous installez l' GuardDuty agent dans vos clusters Amazon EKS, GuardDuty commence à surveiller les événements d'exécution associés à ce cluster. Notez que l' GuardDuty agent et la surveillance du temps d'exécution ne sont pas disponibles pour les nœuds hybrides Amazon EKS. La surveillance du temps d'exécution n'est donc pas disponible pour les événements d'exécution qui se produisent

sur vos nœuds hybrides. S'il GuardDuty identifie une menace potentielle pour votre cluster, il génère un résultat de surveillance du temps d'exécution associé. Par exemple, une menace peut potentiellement commencer par compromettre un conteneur unique qui exécute une application Web vulnérable. Cette application Web peut disposer d'autorisations d'accès aux conteneurs et aux charges de travail sous-jacents. Dans ce scénario, des informations d'identification mal configurées peuvent potentiellement élargir l'accès au compte et aux données qui y sont stockées.

Pour configurer la surveillance du temps d'exécution, vous devez installer l' GuardDuty agent sur votre cluster en tant que module complémentaire Amazon EKS. Pour plus d'informations sur le module complémentaire, consultezthe section called " AWS modules complémentaires".

Pour plus d'informations, consultez la section Runtime Monitoring dans le guide de GuardDuty l'utilisateur Amazon.

Évaluez la résilience du cluster EKS avec AWS Resilience Hub

AWS Resilience Hub évalue la résilience d'un cluster Amazon EKS en analysant son infrastructure. AWS Resilience Hub utilise la configuration du contrôle d'accès basé sur les rôles (RBAC) Kubernetes pour évaluer les charges de travail Kubernetes déployées sur votre cluster. Pour plus d'informations, consultez Activer l'accès de AWS Resilience Hub à votre cluster Amazon EKS dans le Guide de l'utilisateur du AWS Resilience Hub.

Centralisez et analysez les données de sécurité EKS avec Security Lake

Amazon Security Lake est un service de lac de données de sécurité entièrement géré qui vous permet de centraliser les données de sécurité provenant de diverses sources, notamment Amazon EKS. En intégrant Amazon EKS à Security Lake, vous pouvez mieux comprendre les activités effectuées sur vos ressources Kubernetes et améliorer le niveau de sécurité de vos clusters Amazon EKS.

> **Note**
>
> Pour plus d'informations sur l'utilisation de Security Lake avec Amazon EKS et sur la configuration des sources de données, consultez la documentation Amazon Security Lake.

Avantages de l'utilisation de Security Lake avec Amazon Amazon EKS

Données de sécurité centralisées : Security Lake collecte et centralise automatiquement les données de sécurité de vos clusters Amazon EKS, ainsi que les données provenant d'autres AWS services, de fournisseurs de SaaS, de sources sur site et de sources tierces. Cela fournit une vue complète de votre posture de sécurité dans l'ensemble de votre organisation.

Format de données standardisé — Security Lake convertit les données collectées au format Open Cybersecurity Schema Framework (OCSF), qui est un schéma open source standard. Cette normalisation facilite l'analyse et l'intégration avec d'autres outils et services de sécurité.

Détection des menaces améliorée : en analysant les données de sécurité centralisées, notamment les journaux du plan de contrôle Amazon EKS, vous pouvez détecter plus efficacement les activités potentiellement suspectes au sein de vos clusters Amazon EKS. Cela permet d'identifier les incidents de sécurité et d'y répondre rapidement.

Gestion simplifiée des données : Security Lake gère le cycle de vie de vos données de sécurité grâce à des paramètres de rétention et de réplication personnalisables. Cela simplifie les tâches de gestion des données et garantit que vous conservez les données nécessaires à des fins de conformité et d'audit.

Activation de Security Lake pour Amazon EKS

1. Activez la journalisation du plan de contrôle Amazon EKS pour vos clusters EKS. Reportez-vous à la section Activation et désactivation des journaux du plan de contrôle pour obtenir des instructions détaillées.
2. Ajoutez les journaux d'audit Amazon EKS en tant que source dans Security Lake. Security Lake commencera alors à collecter des informations détaillées sur les activités effectuées sur les ressources Kubernetes exécutées dans vos clusters EKS.
3. Configurez les paramètres de rétention et de réplication de vos données de sécurité dans Security Lake en fonction de vos besoins.
4. Utilisez les données OCSF normalisées stockées dans Security Lake pour la réponse aux incidents, les analyses de sécurité et l'intégration avec d'autres AWS services ou outils tiers. Par exemple, vous pouvez générer des informations de sécurité à partir des données Amazon Security Lake à l'aide d'Amazon OpenSearch Ingestion.

Analyse des journaux EKS dans Security Lake

Security Lake normalise les événements du journal EKS au format OCSF, ce qui facilite l'analyse et la corrélation des données avec d'autres événements de sécurité. Vous pouvez utiliser différents outils et services, tels qu'Amazon Athena, Amazon QuickSight ou des outils d'analyse de sécurité tiers, pour interroger et visualiser les données normalisées.

Pour plus d'informations sur le mappage OCSF pour les événements du journal EKS, consultez les journaux d'audit https://github.com/ocsf/examples/tree/main/mappings/markdown/ AWS/v1.1.0/EKS [référence de mappage] dans le référentiel OCSF. GitHub

Activez une connectivité sécurisée entre clusters avec Amazon VPC Lattice

Amazon VPC Lattice est un service de mise en réseau d'applications entièrement géré intégré directement à l'infrastructure AWS réseau que vous pouvez utiliser pour connecter, sécuriser et surveiller vos services sur plusieurs comptes et clouds privés virtuels (). VPCs Avec Amazon EKS, vous pouvez tirer parti d'Amazon VPC Lattice en utilisant le AWS Gateway API Controller, une implémentation de l'API Kubernetes Gateway. À l'aide d'Amazon VPC Lattice, vous pouvez configurer la connectivité entre clusters avec la sémantique standard de Kubernetes de manière simple et cohérente. Pour commencer à utiliser Amazon VPC Lattice avec Amazon EKS, consultez le Guide de l'utilisateur du contrôleur d'API Gateway AWS.

Lancez des clusters EKS à faible latence avec des Zones AWS Locales

Une zone AWS locale est une extension d'une AWS région située à proximité géographique de vos utilisateurs. Les Zones Locales disposent de leurs propres connexions à Internet et prennent en charge AWS Direct Connect. Les ressources créées dans une zone locale peuvent servir les utilisateurs locaux avec des communications à faible latence. Pour plus d'informations, consultez le guide de l'utilisateur des zones AWS locales et le guide de l' EC2 utilisateur Amazon sur les zones locales.

Amazon EKS prend en charge certaines ressources dans Local Zones. Cela inclut les groupes de nœuds gérés, les EC2 nœuds Amazon autogérés, les volumes Amazon EBS et les équilibreurs de charge d'application (). ALBs Nous vous recommandons de tenir compte des points suivants lorsque vous utilisez Local Zones dans le cadre de votre cluster Amazon EKS.

- Vous ne pouvez pas créer de nœuds Fargate dans les Zones Locales avec Amazon EKS.
- Le plan de contrôle Kubernetes géré par Amazon EKS fonctionne toujours dans la région. AWS Le plan de contrôle Kubernetes géré par Amazon EKS ne peut pas fonctionner dans la zone locale. Étant donné que les Local Zones apparaissent comme un sous-réseau au sein de votre VPC, Kubernetes voit vos ressources de zone locale comme faisant partie de ce sous-réseau.
- Le cluster Amazon EKS Kubernetes communique avec les EC2 instances Amazon que vous exécutez dans la AWS région ou la zone locale à l'aide d'interfaces réseau élastiques gérées par Amazon EKS. Pour en savoir plus sur l'architecture réseau Amazon EKS, consultez Configuration de la mise en réseau.
- Contrairement aux sous-réseaux régionaux, Amazon EKS ne peut pas placer d'interfaces réseau dans vos sous-réseaux de zone locale. Cela signifie que vous ne devez pas spécifier de sous-réseaux de Local Zone lorsque vous créez votre cluster. Cependant, vous pouvez avoir des nœuds de travail situés dans différentes Zones Locales connectées au même cluster.

Connectez un cluster Kubernetes à une console de gestion Amazon EKS avec le connecteur Amazon EKS

Vous pouvez utiliser le connecteur Amazon EKS pour enregistrer et connecter tout cluster Kubernetes conforme AWS et le visualiser dans la console Amazon EKS. Une fois qu'un cluster est connecté, vous pouvez voir le statut, la configuration et les charges de travail de ce cluster dans la console Amazon EKS. Vous pouvez utiliser cette fonctionnalité pour afficher les clusters connectés dans la console Amazon EKS, mais vous ne pouvez pas les gérer. Le connecteur Amazon EKS est également un agent qui est un projet open source sur Github. Pour du contenu technique supplémentaire, y compris les questions fréquemment posées et le dépannage, consultezthe section called "Résoudre les problèmes liés au connecteur EKS".

Le connecteur Amazon EKS peut connecter les types de clusters Kubernetes suivants à Amazon EKS.

- Clusters Kubernetes sur site
- Clusters autogérés exécutés sur Amazon EC2
- Clusters gérés par d'autres fournisseurs de cloud

Considérations relatives à Amazon EKS Connector

Avant d'utiliser Amazon EKS Connector, comprenez les points suivants :

- Vous devez disposer des privilèges d'administration sur le cluster Kubernetes pour connecter le cluster à Amazon EKS.
- Le cluster Kubernetes doit comporter des composants master Linux 64 bits (x86) avant de se connecter. Les nœuds de travail ARM ne sont pas pris en charge.
- Vous devez disposer de composants master dans votre cluster Kubernetes qui disposent d'un accès sortant aux points de terminaison Systems Manager `ssm.` et `ssmmessages.`. Pour plus d'informations, consultez la section Points de terminaison de Systems Manager dans le manuel de référence AWS général.
- Par défaut, vous pouvez connecter jusqu'à 10 clusters dans une région. Vous pouvez demander une augmentation par le biais de la console de quota de service. Consultez Demander une augmentation de quota pour plus d'informations.

- Seuls les Amazon EKS`RegisterCluster`, `ListClustersDescribeCluster`, et `DeregisterCluster` APIs sont pris en charge pour les clusters Kubernetes externes.
- Vous devez disposer des autorisations suivantes pour enregistrer un cluster :
  - Eks : RegisterCluster
  - SMS : CreateActivation
  - SMS : DeleteActivation
  - iam : PassRole
- Vous devez disposer des autorisations suivantes pour annuler l'enregistrement d'un cluster :
  - Eks : DeregisterCluster
  - SMS : DeleteActivation
  - SMS : DeregisterManagedInstance

Rôles IAM requis pour Amazon EKS Connector

L'utilisation du Amazon EKS Connector nécessite les deux rôles IAM suivants :

- Le rôle lié à un service du Amazon EKS Connector est créé lorsque vous enregistrez un cluster pour la première fois.
- Vous devez créer le rôle IAM de l'agent Amazon EKS Connector. Consultez the section called "Rôle IAM du connecteur" pour plus de détails.

Pour activer l'autorisation d'affichage du cluster et de la charge de travail pour des principaux IAM, appliquez le `eks-connector` et les rôles de cluster Amazon EKS Connector à votre cluster. Suivez les étapes décrites dans Autoriser l'accès pour afficher les ressources du cluster Kubernetes sur une console Amazon EKS.

Connectez un cluster Kubernetes externe à la console de gestion Amazon EKS

Vous pouvez connecter un cluster Kubernetes externe à Amazon EKS en utilisant plusieurs méthodes dans le cadre du processus suivant. Ce processus comporte deux étapes : enregistrement du cluster auprès d'Amazon EKS et installation de l'agent `eks-connector` dans le cluster.

> **Important**
>
> Vous devez effectuer la deuxième étape dans les 3 jours suivant la fin de la première étape, avant l'expiration de l'enregistrement.

Considérations

Vous pouvez utiliser des manifestes YAML lors de l'installation de l'agent. Vous pouvez également utiliser Helm si vous enregistrez le cluster avec l'interface de ligne de AWS commande AWS Management Console ou. Cependant, vous ne pouvez pas utiliser Helm pour installer l'agent si vous enregistrez le cluster auprès de `eksctl`.

Prérequis

- Assurez-vous que le rôle d'agent Amazon EKS Connector est créé. Suivez les étapes décrites dans Création du rôle d'agent du connecteur Amazon EKS.
- Vous devez disposer des autorisations suivantes pour enregistrer un cluster :
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

Étape 1 : Enregistrement du cluster

Pour enregistrer un cluster sur le connecteur Amazon EKS, vous pouvez utiliser l'un des outils suivants :

- the section called "AWS CLI"
- the section called "AWS Management Console"
- the section called "`eksctl`"

AWS CLI

1. AWS La CLI doit être installée. Pour l'installer ou le mettre à niveau, reportez-vous à la section Installation de la AWS CLI.

2. Pour la configuration du connecteur, spécifiez votre rôle IAM d'agent Amazon EKS Connector. Pour de plus amples informations, veuillez consulter the section called "Rôles IAM requis pour Amazon EKS Connector".

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws: iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

L'exemple qui suit illustre un résultat.

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws: eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws: iam::111122223333:role/AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

Vous utilisez les valeurs `aws-region`, `activationId` et `activationCode` à l'étape suivante.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Sélectionnez Add cluster (Ajouter un cluster), puis Register (Enregistrer) pour afficher la page de configuration.
3. Sur la page Configurer le cluster, renseignez les champs suivants :
   - Nom : nom unique de votre cluster.

- Provider (Fournisseur) : sélectionnez cet élément pour afficher la liste déroulante des fournisseurs de cluster Kubernetes. Si vous ne connaissez pas le fournisseur en question, sélectionnez Autre.
- Rôle EKS Connector : sélectionnez le rôle à utiliser pour connecter le cluster.

4. Sélectionnez Register cluster (Enregistrer le cluster).
5. La page Présentation du cluster s'affiche. Si vous souhaitez utiliser les Charts de Helm, copiez la commande `helm install` et passez à l'étape suivante. Si vous souhaitez utiliser le manifeste YAML, sélectionnez Télécharger le fichier YAML pour télécharger le fichier manifeste sur votre lecteur local.

> **Important**
>
> C'est le seul moment où vous pouvez copier la commande `helm install` ou télécharger ce fichier. Ne quittez pas cette page, car le lien ne sera pas accessible. Vous devez annuler l'enregistrement du cluster et recommencer les étapes depuis le début.

La commande ou le fichier manifeste ne peut être utilisé(e) qu'une seule fois pour le cluster enregistré. Si vous supprimez des ressources du cluster Kubernetes, vous devez réenregistrer le cluster et obtenir un nouveau fichier manifeste.

Passez à l'étape suivante pour appliquer le fichier manifeste à votre cluster Kubernetes.

`eksctl`

1. `eksctl` version `0.68` ou une version ultérieure doit être installée. Pour l'installer ou la mettre à niveau, veuillez consulter la rubrique the section called "Créer un cluster (eksctl)".
2. Enregistrez le cluster en fournissant un nom, un fournisseur et une région.

```
eksctl register cluster --name my-cluster --provider my-provider --region region-code
```

Exemple de sortie :

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
```

```
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-connector-
console-dashboard-full-access-group.yaml" give full EKS Console access to IAM
 identity "<aws-arn>", edit if required; read https://eksctl.io/usage/eks-connector
 for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

Cela crée des fichiers sur votre ordinateur local. Ces fichiers doivent être appliqués au cluster externe dans un délai de 3 jours, sinon l'enregistrement expire.

3. Dans un terminal pouvant accéder au cluster, appliquez le fichier `eks-connector-binding.yaml` :

```
kubectl apply -f eks-connector-binding.yaml
```

Étape 2 : Installation de l'agent **`eks-connector`**

Pour installer l'`eks-connector`agent, utilisez l'un des outils suivants :

- the section called "Helm"
- the section called "yaml"

Helm

> **Note**
>
> Si vous avez enregistré le cluster auprès de`eksctl`, utilisez la méthode du manifeste YAML au lieu de la méthode du graphique Helm.

1. Si vous avez utilisé la AWS CLI à l'étape précédente, remplacez les valeurs `ACTIVATION_CODE` et `ACTIVATION_ID` dans la commande suivante par les `activationCode` valeurs`activationId`, et respectivement. Remplacez le `aws-region` par la AWS région que

vous avez utilisée à l'étape précédente. Ensuite, exécutez la commande pour installer l'agent `eks-connector` dans le cluster d'enregistrement :

```
$ helm install eks-connector \
  --namespace eks-connector \
  oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --set eks.activationCode=ACTIVATION_CODE \
  --set eks.activationId=ACTIVATION_ID \
  --set eks.agentRegion=aws-region
```

Si vous avez utilisé le AWS Management Console à l'étape précédente, utilisez la commande que vous avez copiée à l'étape précédente et qui contient ces valeurs.

2. Vérifiez l'état de santé du déploiement du `eks-connector` installé et attendez que le statut du cluster enregistré dans Amazon EKS soit `ACTIVE`.

yaml

Effectuez la connexion en appliquant le fichier manifeste Amazon EKS Connector à votre cluster Kubernetes. Pour ce faire, vous devez utiliser les méthodes décrites précédemment. Si le manifeste n'est pas appliqué dans les trois jours, l'enregistrement du connecteur Amazon EKS expire. Si la connexion de cluster expire, le cluster doit être désenregistré avant de reconnecter le cluster.

1. Téléchargez le fichier YAML Amazon EKS Connector.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
```

2. Modifiez le fichier YAML Amazon EKS Connector pour remplacer toutes les références de `%AWS_REGION%`, `%EKS_ACTIVATION_ID%`, `%EKS_ACTIVATION_CODE%` par la `aws-region`, l'`activationId` et le `activationCode` à partir de la sortie de l'étape antérieure.

   L'exemple de commande suivant peut remplacer ces valeurs.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-connector.yaml
```

> **Important**
>
> Assurez-vous que votre code d'activation est au format base64.

3. Dans un terminal pouvant accéder au cluster, vous pouvez appliquer le fichier manifeste mis à jour en exécutant la commande suivante :

```
kubectl apply -f eks-connector.yaml
```

4. Après l'application du manifeste Amazon EKS Connector et des fichiers YAML de liaison de rôle à votre cluster Kubernetes, vérifiez que le cluster est maintenant connecté.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

   La sortie doit inclure `status=ACTIVE`.

5. (Facultatif) Attribuez des balises à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ".

Étapes suivantes

Si vous rencontrez des problèmes lors de ces étapes, consultez la rubrique the section called "Résoudre les problèmes liés au connecteur EKS".

Pour accorder à des principaux IAM supplémentaires l'accès à la console Amazon EKS afin de consulter les ressources Kubernetes dans un cluster connecté, consultez. the section called "Accorder l'accès aux clusters"

Autoriser l'accès à l'affichage des ressources du cluster Kubernetes sur une console Amazon EKS

Accordez aux principaux IAM l'accès à la console Amazon EKS pour consulter les informations relatives aux ressources Kubernetes exécutées sur votre cluster connecté.

Prérequis

Le principal IAM que vous utilisez pour accéder AWS Management Console doit répondre aux exigences suivantes :

- Il doit bénéficier de l'autorisation IAM d'`eks:AccessKubernetesApi`.
- Le compte de service Amazon EKS Connector peut emprunter l'identité du principal IAM dans le cluster. Cela permet au connecteur Amazon EKS de mapper le principal IAM à un utilisateur Kubernetes.

Pour créer et appliquer le rôle de cluster Amazon EKS Connector

1. Téléchargez le modèle de rôle de cluster `eks-connector`.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. Modifiez le fichier YAML du modèle de rôle de cluster. Remplacez les références de `%IAM_ARN%` par l'Amazon Resource Name (ARN) de votre principal IAM.
3. Appliquez le rôle de cluster Amazon EKS Connector YAML à votre cluster Kubernetes.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

Pour qu'un directeur IAM puisse consulter les ressources Kubernetes dans la console Amazon EKS, il doit être associé à un Kubernetes `role` ou disposer des autorisations nécessaires pour lire `clusterrole` les ressources. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Pour configurer un principal IAM afin d'accéder au cluster connecté

1. Vous pouvez télécharger n'importe lequel de ces exemples de fichier manifeste pour créer un `clusterrole` et une `clusterrolebinding` ou un `role` et une `rolebinding` :

   Afficher les ressources Kubernetes dans tous les espaces de noms

   - Le rôle de cluster `eks-connector-console-dashboard-full-access-clusterrole` donne accès à tous les espaces de noms et toutes les ressources qui peuvent être visualisés dans la console. Vous pouvez changer le nom du rôle `role`, du

`clusterrole` et de leur liaison correspondante avant de l'appliquer à votre cluster. Utilisez la commande suivante pour télécharger un exemple de fichier.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
eks-connector-console-roles/eks-connector-console-dashboard-full-access-
group.yaml
```

Afficher les ressources Kubernetes dans un espace de noms spécifique

- L'espace de nom dans ce fichier est `default`, donc si vous souhaitez spécifier un espace de noms différent, modifiez le fichier avant de l'appliquer à votre cluster. Utilisez la commande suivante pour télécharger un exemple de fichier.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-
group.yaml
```

2. Modifiez le fichier YAML à accès complet ou à accès restreint pour remplacer les références de `%IAM_ARN%` par l'Amazon Resource Name (ARN) de votre principal IAM.
3. Appliquez les fichiers YAML à accès complet ou à accès restreint à votre cluster Kubernetes. Remplacez la valeur du fichier YAML par la vôtre.

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

Pour afficher les ressources Kubernetes dans votre cluster connecté, consultez the section called "Accès aux ressources du cluster". Les données de certains types de ressources figurant dans l'onglet Ressources ne sont pas disponibles pour les clusters connectés.

Désenregistrer un cluster Kubernetes depuis la console Amazon EKS

Si vous n'utilisez plus un cluster, vous pouvez annuler son enregistrement. Une fois désenregistré, le cluster n'est plus visible dans la console Amazon EKS.

Vous devez disposer des autorisations suivantes pour appeler l'API deregisterCluster :

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`

- `ssm:DeregisterManagedInstance`

Ce processus comporte deux étapes : annulation de l'enregistrement du cluster auprès d'Amazon EKS et désinstallation de l'agent eks-connector dans le cluster.

Désenregistrer le cluster Kubernetes

Pour désenregistrer un cluster du connecteur Amazon EKS, vous pouvez utiliser l'un des outils suivants :

- the section called "AWS CLI"
- the section called "AWS Management Console"
- the section called "`eksctl`"

AWS CLI

1. AWS La CLI doit être installée. Pour l'installer ou le mettre à niveau, reportez-vous à la section Installation de la AWS CLI.
2. Assurez-vous que le rôle d'agent Amazon EKS Connector est créé.
3. Annulez l'enregistrement du cluster connecté.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez Clusters.
3. Sur la page Clusters, sélectionnez le cluster connecté et sélectionnez Deregister (Annuler l'enregistrer).
4. Confirmez que vous voulez annuler l'enregistrement du cluster.

`eksctl`

1. Installez `eksctl` la version `0.68` ou une version ultérieure. Pour l'installer ou la mettre à niveau, veuillez consulter la rubrique the section called "Créer un cluster (eksctl)".
2. Assurez-vous que le rôle d'agent Amazon EKS Connector est créé.
3. Désenregistrez le cluster connecté :

```
eksctl deregister cluster --name my-cluster
```

Nettoyez les ressources de votre cluster Kubernetes

Pour désinstaller l'`eks-connector`agent, utilisez l'un des outils suivants :

- the section called "barre"
- the section called "yaml"

barre

Exécutez la commande suivante pour désinstaller l'agent.

```
helm -n eks-connector uninstall eks-connector
```

yaml

1. Supprimez le fichier YAML Amazon EKS Connector de votre cluster Kubernetes.

```
kubectl delete -f eks-connector.yaml
```

2. Si vous avez créé `clusterrole` ou souhaitez que `clusterrolebindings` d'autres principaux IAM accèdent au cluster, supprimez-les de votre cluster Kubernetes.

Résoudre les problèmes liés au connecteur Amazon EKS

Cette rubrique couvre certaines des erreurs les plus courantes que vous pouvez rencontrer lors de l'utilisation d'Amazon EKS Connector, y compris des instructions sur la façon de les résoudre et des solutions de contournement.

Dépannage de base

Cette section décrit les étapes à suivre pour diagnostiquer les problèmes liés au connecteur Amazon EKS.

Vérifier le statut d'Amazon EKS Connector

Pour vérifier l'état du connecteur Amazon EKS, tapez :

```
kubectl get pods -n eks-connector
```

Inspecter les journaux d'Amazon EKS Connector

Le pod Amazon EKS Connector se compose de trois conteneurs. Pour récupérer les journaux complets de tous ces conteneurs afin que vous puissiez les inspecter, exécutez les commandes suivantes :

- `connector-init`

  ```
  kubectl logs eks-connector-0 --container connector-init -n eks-connector
  kubectl logs eks-connector-1 --container connector-init -n eks-connector
  ```

- `connector-proxy`

  ```
  kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
  kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
  ```

- `connector-agent`

  ```
  kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
  var/log/amazon/ssm/amazon-ssm-agent.log
  kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
  var/log/amazon/ssm/amazon-ssm-agent.log
  ```

Obtenir le nom effectif du cluster

Les clusters Amazon EKS sont identifiés de manière unique au `clusterName` sein d'un seul AWS compte et d'une seule AWS région. Si vous avez plusieurs clusters connectés dans Amazon EKS, vous pouvez confirmer dans quel cluster Amazon EKS le cluster Kubernetes actuel est enregistré. Pour ce faire, saisissez la commande suivante pour connaître le `clusterName` du cluster actuel.

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

Commandes diverses

Les commandes suivantes sont utiles pour récupérer les informations dont vous avez besoin pour résoudre les problèmes.

- Utilisez la commande suivante pour rassembler les images utilisées par les pods dans Amazon EKS Connector.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
   | tr -s '[[:space:]]' '\n'
  ```

- Utilisez la commande suivante pour rassembler les noms des nœuds sur lesquels Amazon EKS Connector est exécuté.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
   '[[:space:]]' '\n'
  ```

- Exécutez la commande suivante pour obtenir les versions de votre client et serveur Kubernetes.

  ```
  kubectl version
  ```

- Exécutez la commande suivante pour obtenir des informations sur vos nœuds.

  ```
  kubectl get nodes -o wide --show-labels
  ```

Problème Helm : 403 Forbidden

Si vous avez reçu le message d'erreur suivant lors de l'exécution des commandes d'installation de Helm :

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

Vous pouvez exécuter la ligne suivante pour y remédier :

```
docker logout public.ecr.aws
```

Erreur de console : le cluster est bloqué dans l'état En attente

Si le cluster reste bloqué dans son `Pending` état sur la console Amazon EKS une fois que vous l'avez enregistré, cela peut être dû au fait que le connecteur Amazon EKS n'a pas AWS encore réussi à connecter le cluster au cluster. Pour un cluster enregistré, l'`Pending`état signifie que la connexion n'est pas correctement établie. Pour résoudre ce problème, assurez-vous d'avoir appliqué le manifeste au cluster Kubernetes cible. Si vous l'avez appliqué au cluster mais que celui-ci est toujours à l'état `Pending`, il est fort probable que le statefulset `eks-connector` ne soit pas sain. Pour résoudre ce problème, consultez the section called "Les pods du connecteur Amazon EKS fonctionnent en boucle" dans cette rubrique.

Erreur de console : l'utilisateur system:serviceaccount:eks-connector:eks-connector ne peut pas se faire passer pour les utilisateurs de ressources du groupe d'API à la portée du cluster

Le connecteur Amazon EKS utilise l'usurpation d'identité d'un utilisateur Kubernetes pour agir au nom des responsables IAM du. AWS Management Console Chaque principal qui accède à l'API Kubernetes depuis le compte de AWS `eks-connector` service doit être autorisé à se faire passer pour l'utilisateur Kubernetes correspondant avec un ARN IAM comme nom d'utilisateur Kubernetes. Dans les exemples suivants, l'ARN IAM est mappé à un utilisateur Kubernetes.

- L'utilisateur IAM *`john`* du AWS compte *`111122223333`* est mappé à un utilisateur Kubernetes. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs.

  ```
  arn:aws: iam::111122223333:user/john
  ```

- Le rôle IAM *`admin`* du AWS compte *`111122223333`* est mappé à un utilisateur Kubernetes :

  ```
  arn:aws: iam::111122223333:role/admin
  ```

  Le résultat est un ARN de rôle IAM, au lieu de l'ARN de session AWS STS.

Pour savoir comment configurer `ClusterRole` et `ClusterRoleBinding` afin d'accorder au compte de service `eks-connector` le privilège de compte de service pour usurper l'identité de l'utilisateur mappé, consultez the section called "Accorder l'accès aux clusters". Assurez-vous qu'il `%IAM_ARN%` est remplacé dans le modèle par l'ARN IAM du principal AWS Management Console IAM.

Erreur de console : [...] est interdit : l'utilisateur [...] ne peut pas répertorier la ressource [...] dans le groupe d'API à l'échelle du cluster

Considérons le problème suivant. Le connecteur Amazon EKS s'est fait passer pour le principal AWS Management Console IAM demandeur dans le cluster Kubernetes cible. Cependant, le principal usurpé n'a pas l'autorisation RBAC pour les opérations de l'API Kubernetes.

Pour résoudre ce problème, il existe deux méthodes pour accorder des autorisations à des utilisateurs supplémentaires. Si vous avez déjà installé eks-connector via les charts de Helm, vous pouvez facilement accorder l'accès aux utilisateurs en exécutant la commande suivante. Remplacez le `userARN1` et `userARN2` par une liste ARNs des rôles IAM permettant d'accéder aux ressources Kubernetes :

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

Ou, en tant qu'administrateur du cluster, accordez le niveau approprié de privilèges RBAC aux utilisateurs individuels de Kubernetes. Pour plus d'informations et d'exemples, consultez the section called "Accorder l'accès aux clusters".

Erreur de console : Amazon EKS ne peut pas communiquer avec le serveur API de votre cluster Kubernetes. Le cluster doit être dans un état ACTIF pour que la connexion soit réussie. Veuillez réessayer dans quelques minutes.

Si le service Amazon EKS ne parvient pas à communiquer avec le connecteur Amazon EKS dans le cluster cible, cela peut être dû à l'une des raisons suivantes :

- Amazon EKS Connector dans le cluster cible n'est pas sain.
- Mauvaise connectivité ou interruption de la connexion entre le cluster cible et la AWS région.

Pour résoudre ce problème, consultez les journaux d'Amazon EKS Connector. Si aucune erreur ne s'affiche concernant le connecteur Amazon EKS, réessayez de vous connecter après quelques minutes. Si vous rencontrez régulièrement une latence élevée ou une connectivité intermittente pour le cluster cible, envisagez de réenregistrer le cluster AWS dans une région située plus près de chez vous.

Les pods du connecteur Amazon EKS fonctionnent en boucle

Un pod de connecteur Amazon EKS peut entrer dans le `CrashLoopBackOff` statut pour de nombreuses raisons. Ce problème concerne probablement le conteneur `connector-init`. Vérifiez l'état du module de connexion Amazon EKS.

```
kubectl get pods -n eks-connector
```

L'exemple qui suit illustre un résultat.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

Si votre sortie est similaire à la sortie précédente, consultez la section the section called "Inspecter les journaux d'Amazon EKS Connector" pour résoudre le problème.

Impossible de lancer le connecteur EKS : InvalidActivation

Lorsque vous démarrez Amazon EKS Connector pour la première fois, il enregistre un `activationId` et `activationCode` avec Amazon Web Services. L'enregistrement peut échouer, ce qui peut provoquer le crash du conteneur `connector-init` avec une erreur similaire à la suivante.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

Pour résoudre ce problème, tenez compte des causes suivantes et des correctifs recommandés :

- L'enregistrement a peut-être échoué car les `activationId` et `activationCode` ne figurent pas dans votre fichier manifeste. Si c'est le cas, assurez-vous qu'il s'agit des valeurs correctes renvoyées par l'opération d'API `RegisterCluster` et que `activationCode` se trouve dans le fichier manifeste. Le `activationCode` est ajouté aux secrets Kubernetes, il doit donc être

encodé. `base64` Pour de plus amples informations, veuillez consulter the section called "Étape 1 : Enregistrement du cluster".

- L'enregistrement a peut-être échoué car votre activation a expiré. En effet, pour des raisons de sécurité, vous devez activer Amazon EKS Connector dans les trois jours suivant l'enregistrement du cluster. Pour résoudre ce problème, assurez-vous que le manifeste du connecteur Amazon EKS est appliqué au cluster Kubernetes cible avant la date et l'heure d'expiration. Pour confirmer la date d'expiration de votre activation, exécutez un appel à l'opération API `DescribeCluster`.

```
aws eks describe-cluster --name my-cluster
```

Dans l'exemple de réponse suivant, la date et l'heure d'expiration sont enregistrées sous la forme `2021-11-12T22:28:51.101000-08:00`.

```
{
    "cluster": {
        "name": "my-cluster",
        "arn": "arn:aws: eks:region:111122223333:cluster/my-cluster",
        "createdAt": "2021-11-09T22:28:51.449000-08:00",
        "status": "FAILED",
        "tags": {
        },
        "connectorConfig": {
            "activationId": "00000000-0000-0000-0000-000000000000",
            "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
            "provider": "OTHER",
            "roleArn": "arn:aws: iam::111122223333:role/my-connector-role"
        }
    }
}
```

Si l'`activationExpiry` est passée, désenregistrez le cluster et enregistrez-le à nouveau. Cela génère une nouvelle activation.

Le nœud du cluster manque de connectivité sortante

Pour fonctionner correctement, le connecteur Amazon EKS nécessite une connectivité sortante vers plusieurs points de AWS terminaison. Vous ne pouvez pas connecter un cluster privé sans connectivité sortante à une AWS région cible. Pour résoudre ce problème, vous devez ajouter la

connectivité sortante nécessaire. Pour plus d'informations sur les exigences de connecteur, consultez the section called "Considérations relatives à Amazon EKS Connector".

Les pods du connecteur Amazon EKS sont en bon `ImagePullBackOff` état

Si vous exécutez la `get pods` commande et que les pods sont en bon `ImagePullBackOff` état, ils ne peuvent pas fonctionner correctement. Si les pods de connecteur Amazon EKS sont en bon `ImagePullBackOff` état, ils ne peuvent pas fonctionner correctement. Vérifiez l'état de vos pods de connecteur Amazon EKS.

```
kubectl get pods -n eks-connector
```

L'exemple qui suit illustre un résultat.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

Le fichier manifeste Amazon EKS Connector par défaut fait référence aux images de la galerie publique Amazon ECR. Il est possible que le cluster Kubernetes cible ne puisse pas extraire les images de la galerie publique Amazon ECR. Veuillez soit résoudre le problème d'extraction d'images de la galerie publique Amazon ECR, soit envisager la mise en miroir des images dans le registre de conteneurs privé de votre choix.

AWS Questions fréquemment posées sur le connecteur

Q : Comment fonctionne la technologie sous-jacente à Amazon EKS Connector ?

R : Le connecteur Amazon EKS est basé sur l'agent AWS Systems Manager (Systems Manager). Le connecteur Amazon EKS s'exécute en tant que connecteur `StatefulSet` sur votre cluster Kubernetes. Il établit une connexion et proxie la communication entre le serveur API de votre cluster et Amazon Web Services. Cela permet d'afficher les données du cluster dans la console Amazon EKS jusqu'à ce que vous déconnectiez le cluster de AWS. L'agent Systems Manager est un projet open source. Pour plus d'informations sur ce projet, consultez la page du projet sur GitHub .

Q : J'ai un cluster Kubernetes sur site que je souhaite connecter. Dois-je ouvrir des ports de pare-feu pour le connecter ?

R : Non, il n'est pas nécessaire d'ouvrir les ports du pare-feu. Le cluster Kubernetes nécessite uniquement une connexion sortante aux régions. AWS AWS les services n'accèdent jamais aux ressources de votre réseau local. Le connecteur Amazon EKS s'exécute sur votre cluster et établit la connexion à AWS. Lorsque l'enregistrement du cluster est terminé, les commandes AWS ne sont envoyées au connecteur Amazon EKS qu'après avoir lancé une action depuis la console Amazon EKS qui nécessite des informations du serveur d'API Kubernetes de votre cluster.

Q : Quelles données sont envoyées depuis mon cluster AWS par le connecteur Amazon EKS ?

R : Le connecteur Amazon EKS envoie les informations techniques nécessaires à l'enregistrement de votre cluster AWS. Il envoie également des métadonnées de cluster et de charge de travail pour les fonctions de la console Amazon EKS demandées par les clients. Amazon EKS Connector ne recueille ou n'envoie ces données que si vous lancez une action à partir de la console Amazon EKS ou de l'API Amazon EKS qui nécessite l'envoi de ces données à AWS. Hormis le numéro de version de Kubernetes, aucune donnée AWS n'est stockée par défaut. Il ne stocke les données que si vous l'y autorisez.

Q : Puis-je connecter un cluster en dehors d'une AWS région ?

R : Oui, vous pouvez connecter un cluster depuis n'importe quel emplacement à Amazon EKS. De plus, votre service Amazon EKS peut être situé dans n'importe quelle AWS région commerciale AWS publique. Cela fonctionne avec une connexion réseau valide entre votre cluster et la AWS région cible. Nous vous recommandons de choisir AWS la région la plus proche de l'emplacement de votre cluster pour optimiser les performances de l'interface utilisateur. Par exemple, si vous avez un cluster en cours d'exécution à Tokyo, connectez-le à la AWS région de Tokyo (c'est-à-dire la `ap-northeast-1` AWS région) pour une faible latence. Vous pouvez connecter un cluster depuis n'importe quel emplacement à Amazon EKS dans toutes les AWS régions commerciales publiques, à l'exception de la Chine ou GovCloud AWS des régions.

Comprendre la sécurité dans Amazon EKS Connector

Amazon EKS Connector est un composant open source qui s'exécute sur votre cluster Kubernetes. Ce cluster peut être situé en dehors de l' AWS environnement. Cela crée des considérations supplémentaires pour les responsabilités en matière de sécurité. Cette configuration peut être illustrée par le schéma suivant. L'orange représente AWS les responsabilités et le bleu représente les responsabilités du client :

Cette rubrique décrit les différences entre le modèle de responsabilité si le cluster connecté est externe à AWS.

AWS responsabilités

- Maintenance, création et fourniture du connecteur Amazon EKS, un composant open source qui s'exécute sur le cluster Kubernetes d'un client et avec lequel il communique. AWS
- Maintien de la sécurité des communications au niveau du transport et de la couche application entre le cluster Kubernetes connecté et les services. AWS

Responsabilités client

- Sécurité spécifique au cluster Kubernetes, notamment dans les domaines suivants :
  - Les secrets Kubernetes doivent être correctement chiffrés et protégés.
  - Verrouillez l'accès à l'espace de noms `eks-connector`.
- Configurer des autorisations de contrôle d'accès basé sur les rôles (RBAC) pour gérer l'accès des principaux IAM depuis AWS. Pour obtenir des instructions, consultez the section called "Accorder l'accès aux clusters".
- Installation et mise à niveau d'Amazon EKS Connector.
- Maintenir le matériel, des logiciels et de l'infrastructure prenant en charge le cluster Kubernetes connecté.

- Sécurisation de leurs AWS comptes (par exemple, en protégeant les informations d'identification de votre utilisateur root).

Déployez Amazon EKS sur site avec Outposts AWS

Vous pouvez utiliser Amazon EKS pour exécuter des applications Kubernetes sur site sur Outposts. AWS Vous pouvez déployer Amazon EKS sur des Outposts avec :

- Clusters étendus : exécutez le plan de contrôle Kubernetes dans une AWS région et les nœuds de votre avant-poste.
- Clusters locaux : exécutez le plan de contrôle et les nœuds Kubernetes sur votre Outpost.

Pour les deux options de déploiement, le plan de contrôle Kubernetes est entièrement géré par. AWS Vous pouvez utiliser les mêmes outils APIs, consoles et outils Amazon que ceux que vous utilisez dans le cloud pour créer et exécuter Amazon EKS sur Outposts.

Le schéma suivant illustre ces options de déploiement.

Quand utiliser chaque option de déploiement

Les clusters locaux et étendus sont des options de déploiement polyvalentes et peuvent être utilisées pour de nombreuses applications.

Avec les clusters locaux, vous pouvez exécuter l'intégralité du cluster Amazon EKS localement sur des Outposts. Cette option permet de réduire le risque d'interruption des applications qui pourrait résulter de déconnexions temporaires du réseau au cloud. Ces déconnexions du réseau peuvent être causées par des coupures de fibre ou des événements météorologiques. Étant donné que l'ensemble du cluster Amazon EKS s'exécute localement sur les Outposts, les applications restent disponibles. Vous pouvez effectuer des opérations de cluster en cas de déconnexion du réseau au

cloud. Pour de plus amples informations, veuillez consulter the section called "Préparez-vous aux déconnexions". Si vous êtes préoccupé par la qualité de la connexion réseau entre vos Outposts et la AWS région parent et que vous avez besoin d'une haute disponibilité via des déconnexions réseau, utilisez l'option de déploiement du cluster local.

Avec les clusters étendus, vous pouvez économiser de la capacité sur votre avant-poste, car le plan de contrôle Kubernetes fonctionne dans la région parent. AWS Cette option convient si vous pouvez investir dans une connectivité réseau fiable et redondante entre votre avant-poste et la AWS région. La qualité de la connexion réseau est essentielle pour cette option. La façon dont Kubernetes gère les déconnexions réseau entre le plan de contrôle Kubernetes et les nœuds peut entraîner des interruptions de service des applications. Pour plus d'informations sur le comportement de Kubernetes, consultez la section Planification, préemption et expulsion dans la documentation de Kubernetes.

Comparaison des options de déploiement

Le tableau suivant compare les différences entre les deux options.

Fonctionnalité	Cluster étendu	Cluster local
Emplacement du plan de contrôle Kubernetes	AWS Région	Outpost
Compte du plan de contrôle Kubernetes	AWS compte	Votre compte
Disponibilité par région	voir Points de terminaison de service	USA Est (Ohio), USA Est (Virginie du Nord), USA Ouest (Californie du Nord), USA Ouest (Oregon), Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney), Asie-Pacifique (Tokyo), Canada (Centre), Europe (Francfort), Europe (Irlande), Europe (Londres), Moyen-Orient (Bahreïn) et Amérique du Sud (São Paulo)

Fonctionnalité	Cluster étendu	Cluster local
Versions Kubernetes mineures	eks/latest/userguide/kubernetes-versions.html [Versions Amazon EKS prises en charge, type="documentation "].	eks/latest/userguide/kubernetes-versions.html [Versions Amazon EKS prises en charge, type="documentation "].
Versions de plateforme	Voir le eks/latest/userguide/platform fichier -versions.html [versions de la plateforme EKS, type="documentation "]	Consultez the section called "Versions de la plateforme EKS"
Facteurs de forme Outpost	Racks Outpost	Racks Outpost
Interfaces utilisateur	AWS Management Console, AWS CLI, `eksctl` API Amazon EKS et Terraform AWS CloudFormation	AWS Management Console, AWS CLI, `eksctl` API Amazon EKS et Terraform AWS CloudFormation
Politiques gérées	Amazon EKSCluster Policy et the section called "AWS politique gérée : Amazon EKSService RolePolicy"	Amazon EKSLocal OutpostClusterPolicy et the section called "AWS politique gérée : Amazon EKSLocal OutpostServiceRolePolicy"
VPC et sous-réseaux de cluster	Consultez the section called "Exigences requises pour le VPC et les sous-réseaux"	Consultez the section called "Créer un VPC et des sous-réseaux"
Accès au point de terminaison de cluster	Public, privé ou les deux	Privés uniquement
Authentification du serveur d'API Kubernetes	AWS Identity and Access Management (IAM) et OIDC	IAM et certificats `x.509`
Types de nœud	Autogérés uniquement	Autogérés uniquement
Types de nœuds de calcul	Amazon à EC2 la demande	Amazon à EC2 la demande

Fonctionnalité	Cluster étendu	Cluster local
Types de stockage de nœuds	Amazon EBS gp2 et SSD local NVMe	Amazon EBS gp2 et SSD local NVMe
Amazon EKS optimisé AMIs	Amazon Linux, Windows et Bottlerocket	Amazon Linux uniquement
Versions d'adresses IP	`IPv4` uniquement	`IPv4` uniquement
Modules complémentaires	Modules complémentaires ou autogérés Amazon EKS	Module complémentaire autogéré uniquement
Interface réseau de conteneur s par défaut	Plugin CNI Amazon VPC pour Kubernetes	Plugin CNI Amazon VPC pour Kubernetes
Journaux du plan de contrôle Kubernetes	Amazon CloudWatch Logs	Amazon CloudWatch Logs
Equilibrage de charge	Utilisez le AWS Load Balancer Controller pour configurer des équilibreurs de charge d'application uniquement (pas d'équilibreurs de charge réseau)	Utilisez le AWS Load Balancer Controller pour configurer des équilibreurs de charge d'application uniquement (pas d'équilibreurs de charge réseau)
Chiffrement d'enveloppe des secrets	Consultez the section called "Activer le chiffrement secret"	Non pris en charge
Rôles IAM pour les comptes de service	Consultez the section called "Identifiants auprès de l'IRSA"	Non pris en charge
Résolution des problèmes	Consultez Résolution des problèmes	Consultez the section called "Résolution des problèmes liés à un cluster"

Créez des clusters Amazon EKS locaux sur AWS Outposts pour une haute disponibilité

Vous pouvez utiliser des clusters locaux pour exécuter l'intégralité de votre cluster Amazon EKS localement sur AWS Outposts. Cela permet de réduire le risque d'interruption des applications qui pourrait résulter de déconnexions temporaires du réseau au cloud. Ces déconnexions peuvent être causées par des coupures de fibre ou des événements météorologiques. Comme l'ensemble du cluster Kubernetes s'exécute localement sur Outposts, les applications restent disponibles. Vous pouvez effectuer des opérations de cluster en cas de déconnexion du réseau au cloud. Pour de plus amples informations, veuillez consulter the section called "Préparez-vous aux déconnexions". Le schéma suivant illustre un déploiement de cluster local.

Les clusters locaux sont généralement disponibles pour une utilisation avec les racks Outposts.

AWS Régions prises en charge

Vous pouvez créer des clusters locaux dans les AWS régions suivantes : USA Est (Ohio), USA Est (Virginie du Nord), USA Ouest (Californie du Nord), USA Ouest (Oregon), Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney), Asie-Pacifique (Tokyo), Canada (Centre), Europe (Francfort), Europe (Irlande), Europe (Londres), Moyen-Orient (Bahreïn), et en Amérique du Sud (São Paulo). Pour des informations détaillées sur les fonctions prises en charge, consultez the section called "Comparaison des options de déploiement".

Déployez un cluster Amazon EKS sur AWS Outposts

Cette rubrique fournit une vue d'ensemble des éléments à prendre en compte lors de l'exécution d'un cluster local sur un Outpost. Elle donne également des instructions pour déployer un cluster local sur un Outpost.

> **Important**
>
> - Ces considérations ne sont pas reproduites dans la documentation Amazon EKS associée. Si d'autres rubriques de la documentation Amazon EKS entrent en conflit avec les considérations présentées ici, suivez ces dernières.
> - Ces considérations sont sujettes à modification et peuvent changer fréquemment. Nous vous recommandons donc de consulter régulièrement cette rubrique.
> - De nombreuses considérations sont différentes de celles relatives à la création d'un cluster sur le AWS cloud.

- Les clusters locaux prennent uniquement en charge les racks Outpost. Un seul cluster local peut s'exécuter sur plusieurs racks Outpost physiques qui constituent un seul Outpost logique. Un seul cluster local ne peut pas fonctionner sur plusieurs Outposts logiques. Chaque Outpost logique possède un seul ARN d'Outpost.
- Les clusters locaux exécutent et gèrent le plan de contrôle Kubernetes dans votre compte sur l'Outpost. Vous ne pouvez pas exécuter de charges de travail sur les instances du plan de contrôle Kubernetes ni modifier les composants du plan de contrôle Kubernetes. Ces nœuds sont gérés par le service Amazon EKS. Les modifications apportées au plan de contrôle Kubernetes ne sont pas conservées par le biais d'actions de gestion automatiques d'Amazon EKS, telles que l'application de correctifs.
- Les clusters locaux prennent en charge les modules complémentaires autogérés et les groupes de nœuds Amazon Linux autogérés. Le plug-in Amazon VPC CNI pour Kubernetes, kube-proxy et CoreDNS est automatiquement installé sur les clusters locaux.
- Les clusters locaux nécessitent l'utilisation d'Amazon EBS sur les Outposts. Amazon EBS doit être disponible dans votre Outpost pour le stockage du plan de contrôle Kubernetes. Les Outposts prennent en charge les volumes `gp2` Amazon EBS uniquement.
- Les Kubernetes basés sur Amazon EBS `PersistentVolumes` sont pris en charge à l'aide du pilote Amazon EBS CSI.
- Les instances du plan de contrôle des clusters locaux sont configurées selon une topologie empilée à haute disponibilité. Deux des trois instances du plan de contrôle doivent être saines à tout moment pour maintenir le quorum. Si le quorum est perdu, contactez le AWS support, car certaines actions côté service seront nécessaires pour activer les nouvelles instances gérées.

Prérequis

- Connaissance des options de déploiement des Outposts, de la sélection des types d'instances et des groupes de placement pour les clusters Amazon EKS sur les AWS Outposts en fonction de considérations relatives à la capacité, ainsi que des exigences et considérations relatives aux VPC.
- Un Outpost existant. Pour plus d'informations, consultez What is AWS Outposts.
- L'outil de ligne de commande `kubectl` est installé sur votre ordinateur ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un principal IAM avec des autorisations pour `create` et `describe` un cluster Amazon EKS. Pour plus d'informations, consultez the section called "Création d'un cluster Kubernetes local sur un Outpost" et the section called "Affichage de la liste ou description de tous les clusters".

Lorsqu'un cluster Amazon EKS local est créé, le principal IAM qui crée le cluster est ajouté de manière permanente. Le principal est spécifiquement ajouté à la table d'autorisation Kubernetes RBAC en tant qu'administrateur. Cette entité possède des autorisations `system:masters`. L'identité de cette entité n'est pas visible dans la configuration de votre cluster. Il est donc important de noter l'entité qui a créé le cluster et de veiller à ne jamais le supprimer. Au départ, seul le principal qui a créé le serveur peut appeler le serveur d'API Kubernetes à l'aide de. `kubectl` Si vous utilisez la console pour créer le cluster, assurez-vous que les mêmes informations d'identification IAM figurent dans la chaîne d'identification du AWS SDK lorsque vous exécutez des `kubectl` commandes sur votre cluster. Une fois votre cluster créé, vous pouvez accorder à d'autres principaux IAM l'accès à votre cluster.

Création d'un cluster local Amazon EKS

Vous pouvez créer un cluster local à l'aide des outils suivants décrits dans cette page :

- the section called "`eksctl`"
- the section called "AWS Management Console"

Vous pouvez également utiliser la AWS CLI, l'API Amazon EKS AWS SDKs, AWS CloudFormationou Terraform pour créer des clusters sur Outposts.

`eksctl`

Pour créer un cluster local avec **`eksctl`**

1. Installez la version `0.210.0` ou une version ultérieure de l'outil de ligne de `eksctl` commande sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Copiez les contenus suivants sur votre appareil. Remplacez les valeurs suivantes, puis exécutez la commande modifiée pour créer le fichier `outpost-control-plane.yaml` :
   - *`region-code`*Remplacez-le par la AWS région prise en charge dans laquelle vous souhaitez créer votre cluster.
   - Remplacez *`my-cluster`* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Remplacez *`vpc-ExampleID1`* et *`subnet-ExampleID1`* par le IDs VPC et le sous-réseau existants. Le VPC et le sous-réseau doivent répondre aux exigences de la section Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur les Outposts. AWS
   - Remplacez *`uniqueid`* par l'identifiant de votre avant-poste.
   - Remplacez *`m5.large`* par un type d'instance disponible sur votre Outpost. Avant de choisir un type d'instance, consultez the section called "Considérations relatives à la capacité". Trois instances du plan de contrôle sont déployées. Vous ne pouvez pas modifier ce numéro.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
```

```
metadata:
  name: my-cluster
  region: region-code
  version: "1.33"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws: outposts:region-code:111122223333:outpost/op-
uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

Pour une liste complète de toutes les options et valeurs par défaut disponibles, consultez AWS Outposts Support et schéma de fichier Config `eksctl` dans la documentation.

3. Créez le cluster à l'aide du fichier de configuration créé à l'étape précédente. `eksctl` crée un VPC et un sous-réseau sur votre Outpost pour y déployer le cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

L'approvisionnement de cluster dure plusieurs minutes. Pendant la création du cluster, plusieurs lignes de sortie apparaissent. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> Pour afficher la plupart des options qui peuvent être spécifiées lors de la création d'un cluster avec `eksctl`, utilisez la commande `eksctl create cluster --help`. Pour consulter toutes les options disponibles, vous pouvez utiliser un fichier `config`. Pour plus d'informations, consultez Utilisation des fichiers de configuration et du schéma du fichier

de configuration dans la documentation `eksctl`. Vous pouvez trouver des exemples de fichiers de configuration sur GitHub.

La `eksctl` commande a automatiquement créé une entrée d'accès pour le principal IAM (utilisateur ou rôle) qui a créé le cluster et lui a accordé des autorisations d'administrateur principal sur les objets Kubernetes du cluster. Si vous ne souhaitez pas que le créateur du cluster dispose d'un accès administrateur aux objets Kubernetes du cluster, ajoutez le texte suivant au fichier de configuration précédent : `bootstrapClusterCreatorAdminPermissions: false` (au même niveau que `metadatavpc`, et). `outpost` Si vous avez ajouté cette option, après la création du cluster, vous devez créer une entrée d'accès pour au moins un principal IAM, sinon aucun principal IAM n'aura accès aux objets Kubernetes du cluster.

AWS Management Console

Pour créer votre cluster à l'aide du AWS Management Console

1. Vous avez besoin d'un VPC et d'un sous-réseau existants qui répondent aux exigences d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Créer un VPC et des sous-réseaux".
2. Si vous possédez déjà un rôle IAM de cluster local, ou si vous comptez créer votre cluster avec`eksctl`, vous pouvez ignorer cette étape. Par défaut, `eksctl` crée un rôle pour vous.

   a. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Créez le rôle IAM de cluster Amazon EKS. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-
document file://"eks-local-cluster-role-trust-policy.json"
```

c. Associez la politique gérée Amazon EKS nommée Amazon EKSLocal OutpostClusterPolicy au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. Ouvrez la console Amazon EKS.
4. En haut de l'écran de la console, assurez-vous que vous avez sélectionné une AWS région prise en charge.
5. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
6. Sur la page Configurer le cluster, saisissez ou sélectionnez les valeurs des champs suivants :
   - Emplacement du plan de contrôle Kubernetes — Choisissez Outposts. AWS
   - ID Outpost (ID d'Outpost) – Choisissez l'ID de l'Outpost sur lequel vous souhaitez créer votre plan de contrôle.
   - Instance type (Type d'instance) – Sélectionnez un type d'instance. Seuls les types d'instances disponibles dans votre Outpost sont affichés. Dans la liste déroulante, chaque type d'instance indique le nombre de nœuds pour lesquels le type d'instance est recommandé. Avant de choisir un type d'instance, consultez the section called "Considérations relatives à la capacité". Tous les réplicas sont déployés à l'aide du même type d'instance. Vous ne pouvez pas modifier le type d'instance une fois votre cluster créé. Trois instances du plan de contrôle sont déployées. Vous ne pouvez pas modifier ce numéro.
   - Nom : nom de votre cluster. Il doit être unique dans votre AWS compte. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.

- Version Kubernetes — Choisissez la version de Kubernetes que vous souhaitez utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous devez utiliser une version antérieure.
- Rôle de service de cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé à l'étape précédente pour permettre au plan de contrôle Kubernetes de gérer les ressources. AWS
- Accès administrateur au cluster Kubernetes : si vous souhaitez que le principal IAM (rôle ou utilisateur) qui crée le cluster dispose d'un accès administrateur aux objets Kubernetes du cluster, acceptez la valeur par défaut (autoriser). Amazon EKS crée une entrée d'accès pour le principal IAM et accorde des autorisations d'administrateur du cluster à cette entrée d'accès. Pour plus d'informations sur les entrées d'accès, consultez the section called "Accorder des autorisations".

  Si vous souhaitez qu'un principal IAM différent de celui qui crée le cluster dispose d'un accès administrateur aux objets du cluster Kubernetes, choisissez l'option Interdire. Après la création du cluster, tout principal IAM disposant des autorisations IAM pour créer des entrées d'accès peut ajouter des entrées d'accès pour tous les principaux IAM ayant besoin d'accéder aux objets du cluster Kubernetes. Pour plus d'informations sur les autorisations IAM requises, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service. Si vous choisissez l'option d'interdiction et que vous ne créez aucune entrée d'accès, aucun utilisateur principal IAM n'aura accès aux objets Kubernetes du cluster.
- Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

7. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :
   - VPC – Choisissez un VPC existant. Le VPC doit disposer d'un nombre suffisant d'adresses IP disponibles pour le cluster, pour tous les nœuds et pour les autres ressources Kubernetes que vous souhaitez créer. Votre VPC doit répondre aux exigences et considérations relatives aux VPC.
   - Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Les sous-réseaux que vous choisissez doivent répondre aux exigences de la section Exigences et considérations relatives aux sous-réseaux.
   - Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS. Amazon EKS crée automatiquement un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces

réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS. Si vous choisissez d'ajouter vos propres groupes de sécurité, vous ne pouvez pas modifier ceux que vous choisissez après la création du cluster. Pour que les hôtes sur site puissent communiquer avec le point de terminaison du cluster, vous devez autoriser le trafic entrant provenant du groupe de sécurité du cluster. Pour les clusters qui ne disposent pas de connexion Internet d'entrée et de sortie (également appelés clusters privés), vous devez effectuer l'une des opérations suivantes :

- Ajoutez le groupe de sécurité associé aux points de terminaison d'un VPC requis. Pour plus d'informations sur les points de terminaison requis, voir the section called "Utilisation des points de terminaison de VPC d'interface " la section Accès aux AWS services par sous-réseau.
- Modifiez le groupe de sécurité qu'Amazon EKS a créé pour autoriser le trafic provenant du groupe de sécurité associé aux points de terminaison d'un VPC. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

8. Sur la page Configurer l'observabilité, vous pouvez éventuellement choisir les options de Métriques et de Journalisation du plan de contrôle que vous voulez activer. Par défaut, chaque type de journal est désactivé.
   - Pour plus d'informations sur l'option Prometheus metrics, consultez. the section called "Étape 1 : activer les métriques Prometheus"
   - Pour plus d'informations sur les options de Journalisation du plan de contrôle, consultez the section called "Journaux du plan de contrôle". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.
9. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez Modifier. Lorsque vous êtes satisfait, choisissez Create. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné.

   L'approvisionnement de cluster dure plusieurs minutes.

Afficher votre cluster local Amazon EKS

1. Une fois votre cluster créé, vous pouvez consulter les instances du plan EC2 de contrôle Amazon qui ont été créées.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

L'exemple qui suit illustre un résultat.

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

Chaque instance est rejetée avec `node-role.eks-local.amazonaws.com/control-plane` pour qu'aucune charge de travail ne soit planifiée sur les instances du plan de contrôle. Pour plus d'informations sur les taches, consultez Taints and Tolerations dans la documentation de Kubernetes. Amazon EKS surveille en permanence l'état des clusters locaux. Nous effectuons des actions de gestion automatiques, telles que des correctifs de sécurité et la réparation des instances défectueuses. Lorsque des clusters locaux sont déconnectés du cloud, nous prenons des mesures pour nous assurer que le cluster retrouve un état sain lors de la reconnexion.

2. Si vous avez créé votre cluster à l'aide de `eksctl`, vous pouvez sauter cette étape. `eksctl` complète cette étape pour vous. Activez `kubectl` pour communiquer avec votre cluster en ajoutant un nouveau contexte au fichier `kubectl config`. Pour savoir comment créer et mettre à jour le fichier, consultez the section called "Cluster d'accès avec kubectl".

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

L'exemple qui suit illustre un résultat.

```
Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

3. Pour vous connecter au serveur d'API Kubernetes de votre cluster local, accédez à la passerelle locale du sous-réseau ou connectez-vous depuis le VPC. Pour plus d'informations sur la connexion d'un rack Outpost à votre réseau local, consultez Comment fonctionnent les passerelles locales pour les racks dans le Guide de l'utilisateur d' AWS Outposts. Si vous utilisez le routage VPC direct et que le sous-réseau Outpost dispose d'une route vers votre passerelle locale, les adresses IP privées des instances du plan de contrôle Kubernetes sont automatiquement diffusées sur votre réseau local. Le point de terminaison du serveur d'API Kubernetes du cluster local est

hébergé sur Amazon Route 53 (Route 53). Le point de terminaison du service d'API peut être résolu par des serveurs DNS publics vers les adresses IP privées des serveurs d'API Kubernetes.

Les instances du plan de contrôle Kubernetes des clusters locaux sont configurées avec des interfaces réseau élastiques statiques avec des adresses IP privées fixes qui ne changent pas tout au long du cycle de vie du cluster. Les machines qui interagissent avec le serveur d'API Kubernetes peuvent ne pas être connectées à Route 53 lors des déconnexions réseau. Si tel est le cas, nous recommandons de configurer `/etc/hosts` avec les adresses IP privées statiques pour la poursuite des opérations. Nous vous recommandons également de configurer des serveurs DNS locaux et de les connecter à votre Outpost. Pour plus d'informations, consultez la documentation AWS des Outposts. Exécutez la commande suivante pour confirmer que la communication est établie avec votre cluster.

```
kubectl get svc
```

L'exemple qui suit illustre un résultat.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

4. (Facultatif) Testez l'authentification auprès de votre cluster local lorsqu'il est déconnecté du AWS cloud. Pour obtenir des instructions, veuillez consulter the section called "Préparez-vous aux déconnexions".

Ressources internes

Amazon EKS crée les ressources suivantes sur votre cluster. Les ressources sont destinées à un usage interne d'Amazon EKS. Pour le bon fonctionnement de votre cluster, ne modifiez ni ne modifiez ces ressources.

- Les modules miroir suivants :
  - `aws-iam-authenticator-`*`node-hostname`*
  - `eks-certificates-controller-`*`node-hostname`*
  - `etcd-`*`node-hostname`*
  - `kube-apiserver-`*`node-hostname`*
  - `kube-controller-manager-`*`node-hostname`*
  - `kube-scheduler-`*`node-hostname`*

- Les modules complémentaires autogérés suivants :
  - `kube-system/coredns`
  - `kube-system/kube-proxy`(non créé tant que vous n'avez pas ajouté votre premier nœud)
  - `kube-system/aws-node` (non créé tant que vous n'avez pas ajouté votre premier nœud). Les clusters locaux utilisent le plug-in Amazon VPC CNI pour Kubernetes pour la mise en réseau des clusters. Ne modifiez pas la configuration des instances du plan de contrôle (pods nommés `aws-node-controlplane-*`). Il existe des variables de configuration que vous pouvez utiliser pour modifier la valeur par défaut lorsque le plugin crée de nouvelles interfaces réseau. Pour plus d'informations, consultez la documentation sur GitHub.
- Les services suivants :
  - `default/kubernetes`
  - `kube-system/kube-dns`
- Une politique `PodSecurityPolicy` nommée `eks.system`
- Un rôle `ClusterRole` nommé `eks:system:podsecuritypolicy`
- Un rôle `ClusterRoleBinding` nommé `eks:system`
- Outre le groupe de sécurité du cluster, Amazon EKS crée un groupe de sécurité nommé dans votre AWS compte`eks-local-internal-do-not-use-or-edit-`*`cluster-name`*`-`*`uniqueid` . Ce groupe de sécurité permet au trafic de circuler librement entre les composants Kubernetes exécutés sur les instances du plan de contrôle.

Étapes suivantes recommandées :

- Accordez au principal IAM qui a créé le cluster les autorisations requises pour consulter les ressources Kubernetes dans le AWS Management Console
- Accordez aux entités IAM l'accès à votre cluster. Si vous souhaitez que les entités puissent consulter les ressources Kubernetes dans la console Amazon EKS, accordez-leur les autorisations requises.
- Configurer la journalisation de votre cluster
- Familiarisez-vous avec ce qui se passe en cas de déconnexion du réseau.
- Ajouter des nœuds à votre cluster
- Envisagez de mettre en place un plan de sauvegarde pour votre `etcd`. Amazon EKS ne prend pas en charge la sauvegarde et la restauration automatisées `etcd` des clusters locaux. Pour plus d'informations, consultez la section Sauvegarde d'un cluster etcd dans la documentation de

Kubernetes. Les deux options principales utilisent `etcdctl` pour automatiser la prise d'instantanés ou l'utilisation de la sauvegarde des volumes de stockage Amazon EBS.

Découvrez les versions des plateformes Kubernetes et Amazon EKS pour Outposts AWS

Les versions de la plateforme de cluster locale représentent les capacités du cluster Amazon EKS sur AWS Outposts. Les versions incluent les composants qui s'exécutent sur le plan de contrôle Kubernetes, pour lesquels les indicateurs du serveur d'API Kubernetes sont activés. Ils incluent également la version actuelle du correctif de Kubernetes. Une ou plusieurs versions de plateforme sont associées à chaque version mineure de Kubernetes. Les versions de plateforme pour les différentes versions mineures de Kubernetes sont indépendantes. Les versions de plateforme pour les clusters locaux et les clusters Amazon EKS dans le cloud sont indépendantes.

Lorsqu'une nouvelle version mineure de Kubernetes est disponible pour les clusters locaux, par exemple, la version initiale de la plate-forme pour cette version mineure de Kubernetes commence à. `1.31` `eks-local-outposts.1` Cependant, Amazon EKS publie régulièrement de nouvelles versions de plateforme pour activer de nouveaux paramètres de plan de contrôle Kubernetes et fournir des correctifs de sécurité.

Lorsque de nouvelles versions de plateforme de clusters locaux deviennent disponibles pour une version mineure :

- Le numéro de version de plateforme est incrémenté (`eks-local-outposts.n+1`).
- Amazon EKS met automatiquement à jour tous les clusters locaux existants vers la dernière version de plate-forme pour la version mineure de Kubernetes correspondante. Les mises à jour automatiques de versions de plateforme existantes sont déployées de façon incrémentielle. Le processus de déploiement consiste à remplacer les instances du plan de contrôle Kubernetes gérées exécutées sur l'Outpost, une par une, jusqu'à ce que les 3 instances soient remplacées par de nouvelles.
- Le processus de remplacement de l'instance du plan de contrôle Kubernetes cessera de progresser en cas de risque d'interruption de service. Amazon EKS tentera de remplacer une instance uniquement si les deux autres instances du plan de contrôle Kubernetes sont saines et répondent à toutes les conditions de disponibilité en tant que nœud de cluster.
- Le déploiement d'une version de plateforme prend généralement moins de 30 minutes. Si un cluster reste `UPDATING` actif pendant une période prolongée, consultez le Support the section called "Résolution des problèmes liés à un cluster" et demandez de l' AWS aide. Ne mettez jamais

fin manuellement aux instances du plan de contrôle Kubernetes, sauf indication contraire du Support. AWS

- Amazon EKS peut publier une nouvelle AMI de nœud avec une version de correctif correspondante. Toutes les versions de correctif sont compatibles entre le plan de contrôle et le nœud Kubernetes AMIs pour une seule version mineure de Kubernetes.

Les nouvelles versions de la plateforme n'introduisent pas de modifications majeures ni n'entraînent d'interruptions de service.

Les clusters locaux sont toujours créés avec la dernière version de plate-forme disponible (`eks-local-outposts.n`) pour la version de Kubernetes spécifiée.

Les versions actuelles et récentes de plateforme sont décrites dans les tableaux ci-dessous.

Pour recevoir des notifications concernant toutes les modifications apportées au fichier source sur cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante avec un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/
eks-outposts-platform-versions.adoc.atom
```

Version de Kubernetes **`1.31`**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de la `1.31` plateforme : `CertificateApproval` `CertificateSigningCertificateSubjectRestriction`,`ClusterTrustBundleAttest`,`DefaultIetValidatingAdmissionWebhook`.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.31.6`	`eks-local-outposts.1`	Publication initiale de la version Kubernetes pour les clusters Amazon EKS `v1.31` locaux sur Outposts.	9 avril 2025

Version de Kubernetes **1.30**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de la `1.30` plateforme : `CertificateApproval` `CertificateSigningCertificateSubjectRestriction`,`ClusterTrustBundleAttest`,`DefaultIetValidatingAdmissionWebhook`.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.30.10`	`eks-local-outposts.3`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.30.10` AWS Authentificateur IAM mis à jour vers. `v0.6.29` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.2`. CoreDNS mis à jour vers. `v1.11.4` AWS Cloud Controller Manager a été mis à jour vers`v1.30.8`. Version Bottlerocket mise à jour vers. `v1.34.0`	27 mars 2025
`1.30.7`	`eks-local-outposts.2`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité.	10 janvier 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
		Kube-proxy mis à jour vers. `v1.30.7` AWS Authentificateur IAM mis à jour vers. `v0.6.28` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.0`. Version de Bottlerocket mise à jour vers. `v1.29.0`	
`1.30.5`	`eks-local-outposts.1`	Publication initiale de la version Kubernetes pour les clusters Amazon EKS `v1.30` locaux sur Outposts.	13 novembre 2024

Version de Kubernetes **`1.29`**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de la `1.29` plateforme : `CertificateApproval` `CertificateSigningCertificateSubjectRestriction`,`ClusterTrustBundleAttest`,`DefaultIetValidatingAdmissionWebhook`.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`v1.29.14`	`eks-local-outposts.6`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité.	27 mars 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
		Kube-proxy mis à jour vers. `v1.29.14` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.2`. CoreDNS mis à jour vers. `v1.11.4` AWS Cloud Controller Manager a été mis à jour vers`v1.29.8`. Version Bottlerocket mise à jour vers. `v1.34.0`	
`v1.29.11`	`eks-local-outposts.5`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.29.11` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.0`. Image CoreDNS mise à jour en. `v1.11.3` Version de Bottlerocket mise à jour vers. `v1.29.0`	10 janvier 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.29.9`	`eks-local-outposts.4`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.29.9` AWS Authentificateur IAM mis à jour vers. `v0.6.26` Version de Bottlerocket mise à jour vers. `v1.26.0`	8 novembre 2024
`1.29.6`	`eks-local-outposts.3`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. `v1.22.0`	22 octobre 2024
`1.29.6`	`eks-local-outposts.2`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. `v1.21.0`	27 août 2024
`1.29.6`	`eks-local-outposts.1`	Publication initiale de la version Kubernetes pour les clusters Amazon EKS `v1.29` locaux sur Outposts.	20 août 2024

Version de Kubernetes **1.28**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de la 1.28 plateforme : CertificateApproval CertificateSigningCertificateSubjectRestriction,ClusterTrustBundleAttest,DefaultIetValidatingAdmissionWebhook.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.28.15	eks-local-outposts.13	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Mise à jour de la v1.28.15 version de kube-proxy. Le plugin Amazon VPC CNI pour Kubernetes a été mis à jour vers. v1.19.0 AWS Cloud Controller Manager a été mis à jour versv1.28.11.	27 mars 2025
1.28.15	eks-local-outposts.12	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.28.15 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.19.0. Version de	10 janvier 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
		Bottlerocket mise à jour vers. `v1.29.0`	
`1.28.14`	`eks-local-outposts.11`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.28.14` AWS Authentificateur IAM mis à jour vers. `v0.6.26` Image CoreDNS mise à jour en. `v1.11.1` Version de Bottlerocket mise à jour vers. `v1.26.0`	8 novembre 2024
`1.28.10`	`eks-local-outposts.10`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. `v1.22.0`	22 octobre 2024
`1.28.10`	`eks-local-outposts.9`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. `v1.21.0`	27 août 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.28.10`	`eks-local-outposts.8`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 juillet 2024
`1.28.10`	`eks-local-outposts.6`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.28.10` AWS Authentificateur IAM mis à jour vers. `v0.6.20` Version de Bottlerocket mise à jour vers. `v1.20.2`	19 juin 2024
`1.28.6`	`eks-local-outposts.5`	Mise à jour de la version v1.19.3 de Bottlerocket contenant les dernières corrections de bogues pour prendre en charge le démarrage local dans Outposts.	18 avril 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.28.6`	`eks-local-outposts.4`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Support rétabli ou démarrage local dans Outposts. Version de Bottlerocket rétrogradée à pour des raisons de compatibilité. `v1.15.1`	2 avril 2024
`1.28.6`	`eks-local-outposts.3`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	22 mars 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.28.6`	`eks-local-outposts.2`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.28.6` AWS Authentificateur IAM mis à jour vers. `v0.6.17` Le plug-in Amazon VPC CNI pour Kubernetes a été rétrogradé à pour des raisons de compatibilité. `v1.13.2` Version de Bottlerocket mise à jour vers. `v1.19.2`	8 mars 2024
`1.28.1`	`eks-local-outposts.1`	Publication initiale de la version Kubernetes pour les clusters Amazon EKS `v1.28` locaux sur Outposts.	4 octobre 2023

Version de Kubernetes **`1.27`**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de la `1.27` plateforme : `CertificateApproval` `CertificateSigningCertificateSubjectRestriction`,`ClusterTrustBundleAttest`,`DefaultIetValidatingAdmissionWebhook`.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.27.16`	`eks-local-outposts.13`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.2`. Version Bottlerocket mise à jour vers. `v1.34.0`	27 mars 2025
`1.27.16`	`eks-local-outposts.12`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.0`. Version de Bottlerocket mise à jour vers. `v1.29.0`	10 janvier 2025
`1.27.16`	`eks-local-outposts.11`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.27.16` AWS Authentificateur IAM mis à jour vers. `v0.6.26` Image	8 novembre 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
		CoreDNS mise à jour en. `v1.11.1` Version de Bottlerocket mise à jour vers. `v1.26.0`	
`1.27.14`	`eks-local-outposts.10`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. `v1.22.0`	22 octobre 2024
`1.27.14`	`eks-local-outposts.9`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. `v1.21.0`	27 août 2024
`1.27.14`	`eks-local-outposts.8`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 juillet 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.27.14`	`eks-local-outposts.6`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.27.14` AWS Authentificateur IAM mis à jour vers. `v0.6.20` Version de Bottlerocket mise à jour vers. `v1.20.2`	19 juin 2024
`1.27.10`	`eks-local-outposts.5`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2024
`1.27.10`	`eks-local-outposts.4`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.27.10` AWS Authentificateur IAM mis à jour vers. `v0.6.17` Version de Bottlerocket mise à jour vers. `v1.19.2`	22 mars 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
`1.27.3`	`eks-local-outposts.3`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.27.3` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.13.2`.	14 juillet 2023
`1.27.1`	`eks-local-outposts.2`	Image CoreDNS mise à jour en. `v1.10.1`	22 juin 2023
`1.27.1`	`eks-local-outposts.1`	Publication initiale de la version Kubernetes pour les clusters Amazon EKS `1.27` locaux sur Outposts.	30 mai 2023

Création d'un VPC et de sous-réseaux pour les clusters Amazon EKS sur Outposts AWS

Lorsque vous créez un cluster local, vous spécifiez un VPC et au moins un sous-réseau privé qui fonctionne sur les Outposts. Cette rubrique fournit une vue d'ensemble des exigences et considérations spécifiques au VPC et aux sous-réseaux pour votre cluster local.

Exigences et considérations requises pour le VPC

Lorsque vous créez un cluster local, le VPC que vous spécifiez doit répondre aux exigences et aux considérations suivantes :

- Assurez-vous que le VPC dispose de suffisamment d'adresses IP pour le cluster local, les nœuds et les autres ressources Kubernetes que vous souhaitez créer. Si le VPC que vous souhaitez utiliser ne possède pas suffisamment d'adresses IP, augmentez le nombre d'adresses IP disponibles. Vous pouvez effectuer cette opération en associant des blocs d'adresse CIDR (Routage inter-domaines sans classe) supplémentaires avec votre VPC. Vous pouvez associer des blocs d'adresses CIDR privés (RFC 1918) et publics (non-RFC 1918) à votre VPC avant ou après la création de votre cluster. Un cluster peut prendre jusqu'à 5 heures avant qu'un bloc d'adresse CIDR associé à un VPC ne soit reconnu.
- Le VPC ne peut pas avoir attribué de préfixes IP ou IPv6 de blocs CIDR. En raison de ces contraintes, les informations traitées dans Attribuer plus d'adresses IP aux nœuds Amazon EKS avec des préfixes ne s'appliquent the section called "IPv6" pas à votre VPC.
- Le VPC possède un nom d'hôte DNS et une résolution DNS activés. Sans ces fonctions, la création du cluster local échoue, et vous devez activer les fonctions et recréer votre cluster. Pour plus d'informations, consultez Attributs DNS pour votre VPC dans le guide de l'utilisateur d'Amazon VPC.
- Pour accéder à votre cluster local via votre réseau local, le VPC doit être associé à la table de routage de la passerelle locale de votre Outpost. Pour plus d'informations, consultez la section Associations VPC dans le Guide de l'utilisateur d' AWS Outposts.

Exigences et considérations requises pour les sous-réseaux

Lorsque vous créez le cluster, spécifiez au moins un sous-réseau privé. Si vous spécifiez plusieurs sous-réseaux, les instances du plan de contrôle Kubernetes sont réparties uniformément sur les sous-réseaux. Si plus d'un sous-réseau est spécifié, les sous-réseaux doivent exister sur le même Outpost. En outre, les sous-réseaux doivent également disposer d'un acheminement et d'autorisations de groupe de sécurité appropriés pour communiquer entre eux. Lorsque vous créez un cluster local, les sous-réseaux que vous spécifiez doivent répondre aux exigences suivantes :

- Les sous-réseaux doivent se trouver sur le même Outpost logique.
- Ensemble, les sous-réseaux disposent d'au moins trois adresses IP disponibles pour les instances du plan de contrôle Kubernetes. Si trois sous-réseaux sont spécifiés, chaque sous-réseau doit avoir au moins une adresse IP disponible. Si deux sous-réseaux sont spécifiés, chaque sous-réseau doit avoir au moins deux adresses IP disponibles. Si un sous-réseau est spécifié, le sous-réseau doit avoir au moins trois adresses IP disponibles.
- Les sous-réseaux ont une route vers la passerelle locale du rack Outpost pour accéder au serveur d'API Kubernetes via votre réseau local. Si les sous-réseaux n'ont pas de route vers la passerelle

locale du rack Outpost, vous devez communiquer avec votre serveur d'API Kubernetes depuis le VPC.

- Les sous-réseaux doivent utiliser une dénomination basée sur une adresse IP. La dénomination EC2 basée sur les ressources Amazon n'est pas prise en charge par Amazon EKS.

Accès aux services par sous-réseau AWS

Les sous-réseaux privés du cluster local sur les Outposts doivent être en mesure de communiquer avec AWS les services régionaux. Vous pouvez y parvenir en utilisant une passerelle NAT pour l'accès Internet sortant ou, si vous voulez préserver la confidentialité de tout le trafic au sein de votre VPC, en utilisant des points de terminaison d'un VPC d'interface.

Utiliser une passerelle NAT

Les sous-réseaux privés du cluster local sur les Outposts doivent être associés à une table de routage comportant une route vers une passerelle NAT dans un sous-réseau public situé dans la zone de disponibilité parent de l'Outpost. Le sous-réseau public doit disposer d'un acheminement vers une passerelle Internet. La passerelle NAT permet l'accès à Internet sortant et empêche les connexions entrantes non sollicitées depuis Internet vers les instances de l'Outpost.

Utilisation des points de terminaison de VPC d'interface

Si les sous-réseaux privés du cluster local sur Outposts ne disposent pas d'une connexion Internet sortante, ou si vous souhaitez que tout le trafic reste privé au sein de votre VPC, vous devez créer les points de terminaison VPC d'interface et point de terminaison de passerelle suivants dans un sous-réseau régional avant de créer votre cluster.

Point de terminaison	Type de point de terminaison
`com.amazonaws.` *`region-code`* `.ssm`	utilisateur
`com.amazonaws.` *`region-code`* `.ssmmessages`	utilisateur
`com.amazonaws.` *`region-code`* `.ec2messages`	utilisateur
`com.amazonaws.` *`region-code`* `.ec2`	utilisateur

Point de terminaison	Type de point de terminaison
`com.amazonaws.` *`region-code`* `.secretsmanager`	utilisateur
`com.amazonaws.` *`region-code`* `.logs`	utilisateur
`com.amazonaws.` *`region-code`* `.sts`	utilisateur
`com.amazonaws.` *`region-code`* `.ecr.api`	utilisateur
`com.amazonaws.` *`region-code`* `.ecr.dkr`	utilisateur
`com.amazonaws.` *`region-code`* `.s3`	Passerelle

Les points de terminaison doivent répondre aux critères suivants :

- Créé dans un sous-réseau privé situé dans la zone de disponibilité parent de votre Outpost
- Noms DNS privés activés
- Avoir un groupe de sécurité attaché qui autorise le trafic HTTPS entrant à partir de la plage CIDR du sous-réseau privé d'Outpost.

La création de points de terminaison entraîne des frais. Pour en savoir plus, consultez Pricing AWS PrivateLink (Tarification). Si vos Pods ont besoin d'accéder à d'autres AWS services, vous devez créer des points de terminaison supplémentaires. Pour une liste complète des points de terminaison, consultez la section AWS Services intégrés à AWS PrivateLink.

Création d'un VPC

Vous pouvez créer un VPC répondant aux exigences précédentes à l'aide de l'un des modèles suivants : AWS CloudFormation

- Modèle 1 — Ce modèle crée un VPC avec un sous-réseau privé sur l'avant-poste et un sous-réseau public dans la région. AWS Le sous-réseau privé dispose d'une route vers Internet via une passerelle NAT qui réside dans le sous-réseau public de la AWS région. Ce modèle peut être utilisé pour créer un cluster local dans un sous-réseau avec un accès Internet de sortie.

- Modèle 2 — Ce modèle crée un VPC avec un sous-réseau privé sur l'Outpost et l'ensemble minimal de points de terminaison VPC requis pour créer un cluster local dans un sous-réseau dépourvu d'accès Internet d'entrée ou de sortie (également appelé sous-réseau privé).

Préparez les clusters Amazon EKS locaux sur les AWS Outposts pour les déconnexions réseau

Si votre réseau local n'est plus connecté au AWS cloud, vous pouvez continuer à utiliser votre cluster Amazon EKS local sur un Outpost. Cette rubrique explique comment préparer votre cluster local pour les déconnexions du réseau et les considérations connexes.

- Les clusters locaux garantissent la stabilité et la continuité des opérations lors de déconnexions réseau temporaires et imprévues. AWS Outposts reste une offre entièrement connectée qui agit comme une extension du AWS cloud dans votre centre de données. En cas de déconnexion du réseau entre votre Outpost et AWS Cloud, nous vous recommandons de tenter de rétablir votre connexion. Pour obtenir des instructions, consultez la liste de vérification du dépannage du réseau AWS Outposts Rack dans le guide de l'utilisateur d'Outposts AWS . Pour plus d'informations sur la résolution des problèmes liés aux clusters locaux, consultez the section called “Résolution des problèmes liés à un cluster”.
- Les Outposts génèrent une métrique `ConnectedStatus` qui permet de surveiller l'état de connectivité de votre Outpost. Pour plus d'informations, consultez la section Outposts Metrics dans le guide de l'utilisateur d' AWS Outposts.
- Les clusters locaux utilisent IAM comme mécanisme d'authentification par défaut à l'aide de l'authentificateur AWS Identity and Access Management pour Kubernetes. L'IAM n'est pas disponible lors des déconnexions du réseau. Les clusters locaux prennent en charge un mécanisme d'authentification à l'aide des certificats `x.509` que vous pouvez utiliser pour vous connecter à votre cluster lors des déconnexions du réseau. Pour plus d'informations sur l'obtention et l'utilisation d'un certificat `x.509` pour votre cluster, consultez the section called “Authentification auprès de votre cluster local lors d'une déconnexion du réseau”.
- Si vous ne pouvez pas accéder à Route 53 lors de déconnexions réseau, pensez à utiliser des serveurs DNS locaux dans votre environnement local. Les instances du plan de contrôle Kubernetes utilisent des adresses IP statiques. Vous pouvez configurer les hôtes que vous utilisez pour vous connecter à votre cluster avec le nom d'hôte et les adresses IP du point de terminaison comme alternative à l'utilisation de serveurs DNS locaux. Pour plus d'informations, consultez la section DNS dans le Guide de l'utilisateur d' AWS Outposts.

- Si vous prévoyez une augmentation du trafic d'applications lorsque le réseau se déconnecte, vous pouvez allouer de la capacité de calcul de réserve dans votre cluster lorsque vous êtes connecté au cloud. Les EC2 instances Amazon sont incluses dans le prix d' AWS Outposts. L'exécution d'instances de rechange n'a donc aucun impact sur vos coûts AWS d'utilisation.
- Lors des déconnexions du réseau pour permettre les opérations de création, de mise à jour et de mise à l'échelle des charges de travail, les images de conteneur de votre application doivent être accessibles sur le réseau local et votre cluster doit disposer d'une capacité suffisante. Les clusters locaux n'hébergent pas de registre de conteneurs pour vous. Si les pods ont déjà été exécutés sur ces nœuds, les images des conteneurs sont mises en cache sur les nœuds. Si vous téléchargez généralement les images de conteneurs de votre application depuis Amazon ECR dans le cloud, envisagez d'exécuter un cache ou un registre local. Un cache ou un registre local est utile si vous avez besoin d'opérations de création, de mise à jour et de mise à l'échelle pour les ressources de la charge de travail pendant les déconnexions du réseau.
- Les clusters locaux utilisent Amazon EBS comme classe de stockage par défaut pour les volumes persistants et le pilote CSI Amazon EBS pour gérer le cycle de vie des volumes persistants Amazon EBS. Lors des déconnexions du réseau, les pods soutenus par Amazon EBS ne peuvent pas être créés, mis à jour ou redimensionnés. En effet, ces opérations nécessitent des appels à l'API Amazon EBS dans le cloud. Si vous déployez des charges de travail dynamiques sur des clusters locaux et que vous avez besoin d'opérations de création, de mise à jour ou de dimensionnement lors de déconnexions réseau, envisagez d'utiliser un autre mécanisme de stockage.
- Les instantanés Amazon EBS ne peuvent pas être créés ou supprimés si les AWS Outposts ne peuvent pas accéder à la AWS région concernée APIs (comme pour APIs Amazon EBS ou Amazon S3).
- Lors de l'intégration d'ALB (Ingress) à AWS Certificate Manager (ACM), les certificats sont transmis et stockés dans la mémoire de l'instance AWS Outposts ALB Compute. La terminaison TLS actuelle continuera de fonctionner en cas de déconnexion de la AWS région. Dans ce contexte, les opérations de mutation (telles que les nouvelles définitions d'entrée, les nouvelles opérations d'API de certificats basés sur ACM, le dimensionnement du calcul ALB ou la rotation des certificats) échoueront. Pour plus d'informations, consultez la section Résolution des problèmes liés au renouvellement géré des certificats dans le Guide de l'utilisateur de AWS Certificate Manager.
- Les journaux du plan de contrôle Amazon EKS sont mis en cache localement sur les instances du plan de contrôle Kubernetes lors des déconnexions du réseau. Lors de la reconnexion, les journaux sont envoyés aux CloudWatch journaux de la AWS région parent. Vous pouvez utiliser les solutions partenaires Prometheus, Grafana ou Amazon EKS pour surveiller le cluster localement à

l'aide du point de terminaison des métriques du serveur d'API Kubernetes ou en utilisant Fluent Bit pour les journaux.

- Si vous utilisez le AWS Load Balancer Controller sur Outposts pour le trafic des applications, les pods existants dirigés par le AWS Load Balancer Controller continuent de recevoir du trafic lors des déconnexions du réseau. Les nouveaux pods créés lors des déconnexions du réseau ne reçoivent pas de trafic tant que l'Outpost n'est pas reconnecté au Cloud. AWS Envisagez de définir le nombre de répliques pour vos applications lorsque vous êtes connecté au AWS cloud afin de répondre à vos besoins de mise à l'échelle lors des déconnexions du réseau.
- Le plug-in Amazon VPC CNI pour Kubernetes utilise par défaut le mode IP secondaire. Il est configuré avec `WARM_ENI_TARGET =1`, ce qui permet au plugin de conserver « une interface réseau élastique complète » contenant les adresses IP disponibles. Pensez à modifier les valeurs `WARM_ENI_TARGET`, `WARM_IP_TARGET` et `MINIMUM_IP_TARGET` en fonction de vos besoins de mise à l'échelle lors d'un état déconnecté. Pour plus d'informations, consultez le fichier readme du plugin sur GitHub. Pour obtenir la liste du nombre maximal de pods pris en charge par chaque type d'instance, consultez le fichier eni-max-pods.txt sur GitHub.

Authentification auprès de votre cluster local lors d'une déconnexion du réseau

AWS Identity and Access Management (IAM) n'est pas disponible lors des déconnexions réseau. Vous ne pouvez pas vous authentifier auprès de votre cluster local à l'aide des informations d'identification IAM lorsque vous êtes déconnecté. Cependant, vous pouvez vous connecter à votre cluster sur votre réseau local à l'aide de certificats `x509` en cas de déconnexion. Vous devez télécharger et stocker un certificat `X509` client à utiliser lors des déconnexions. Dans cette rubrique, vous apprendrez à créer et à utiliser le certificat pour vous authentifier auprès de votre cluster lorsqu'il est déconnecté.

1. Créer une demande de signature de certificat.

   a. Générez une demande de signature de certificat.

   ```
   openssl req -new -newkey rsa:4096 -nodes -days 365 \
       -keyout admin.key -out admin.csr -subj "/CN=admin"
   ```

   b. Créez une demande de signature de certificat dans Kubernetes.

   ```
   BASE64_CSR=$(cat admin.csr | base64 -w 0)
   cat << EOF > admin-csr.yaml
   apiVersion: certificates.k8s.io/v1
   ```

```
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. Créez une demande de signature de certificat à l'aide de `kubectl`.

```
kubectl create -f admin-csr.yaml
```

3. Vérifiez le statut de la demande de signature de certificat.

```
kubectl get csr admin-csr
```

L'exemple qui suit illustre un résultat.

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

Kubernetes a créé la demande de signature du certificat.

4. Approuvez la demande de signature de certificat.

```
kubectl certificate approve admin-csr
```

5. Vérifiez à nouveau l'état de la demande de signature de certificat pour approbation.

```
kubectl get csr admin-csr
```

L'exemple qui suit illustre un résultat.

```
NAME       AGE   REQUESTOR                     CONDITION
admin-csr  11m   kubernetes-admin              Approved
```

6. Récupérez et vérifiez le certificat.

   a. Récupérez le certificat.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
 admin.crt
```

b. Vérifiez le certificat.

```
cat admin.crt
```

7. Créez une liaison de rôle de cluster pour un utilisateur `admin`.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. Générez un kubeconfig spécifique à l'utilisateur pour un état déconnecté.

   Vous pouvez générer un fichier `kubeconfig` à l'aide des certificats `admin` téléchargés. Remplacez *`my-cluster`* et *`apiserver-endpoint`* dans les commandes suivantes.

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. Affichez votre fichier `kubeconfig`.

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Si des services sont déjà en production sur votre Outpost, ignorez cette étape. Si Amazon EKS est le seul service exécuté sur votre Outpost et que celui-ci n'est pas actuellement en production,

vous pouvez simuler une déconnexion du réseau. Avant de passer en production avec votre cluster local, simulez une déconnexion pour vous assurer que vous pouvez accéder à votre cluster lorsqu'il est déconnecté.

a. Appliquez des règles de pare-feu sur les appareils réseau qui connectent votre avant-poste à la AWS région. Cela déconnecte le lien de service de l'Outpost. Vous ne pouvez pas créer de nouvelles instances. Les instances en cours d'exécution perdent leur connectivité à la AWS Région et à Internet.

b. Vous pouvez tester la connection à votre cluster local à l'aide du certificat `x509` en cas de déconnexion. Assurez-vous de remplacer votre `kubeconfig` par le `admin.kubeconfig` que vous avez créé à une étape précédente. *`my-cluster`*Remplacez-le par le nom de votre cluster local.

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

Si vous remarquez des problèmes avec vos clusters locaux alors qu'ils sont déconnectés, nous vous recommandons d'ouvrir un ticket d'assistance.

Sélectionnez les types d'instances et les groupes de placement pour les clusters Amazon EKS sur AWS Outposts en fonction de considérations de capacité

Cette rubrique fournit des conseils pour sélectionner le type d'instance du plan de contrôle Kubernetes et (éventuellement) utiliser des groupes de placement pour répondre aux exigences de haute disponibilité de votre cluster Amazon EKS local sur un Outpost.

Avant de sélectionner un type d'instance (tel que `m5c5`, `our5`) à utiliser pour le plan de contrôle Kubernetes de votre cluster local sur Outposts, vérifiez les types d'instance disponibles dans votre configuration Outpost. Après avoir identifié les types d'instance disponibles, sélectionnez la taille de l'instance (telle que `large`, `xlarge` ou `2xlarge`) en fonction du nombre de nœuds requis par vos charges de travail. Le tableau suivant fournit des recommandations pour choisir une taille d'instance.

> **Note**
>
> Les tailles d'instance doivent avoir été définies sur vos Outposts. Assurez-vous que vous avez suffisamment de capacité pour trois instances de la taille disponible sur vos Outposts pour la durée de vie de votre cluster local. Pour obtenir la liste des types d' EC2 instances

Amazon disponibles, consultez les sections Calcul et stockage de la section Fonctionnalités du rack AWS Outposts.

Nombre de modèles	Taille des instances du plan de contrôle Kubernetes
1-20	`large`
21–100	`xlarge`
101–250	`2xlarge`
251–500	`4xlarge`

Le stockage du plan de contrôle Kubernetes nécessite 246 Go de stockage Amazon EBS pour chaque cluster local afin de répondre aux IOPS requises pour. `etcd` Lorsque le cluster local est créé, les volumes Amazon EBS sont alloués automatiquement pour vous.

Placement du plan de contrôle

Lorsque vous ne spécifiez pas de groupe de placement avec la `OutpostConfig.ControlPlanePlacement.GroupName` propriété, les EC2 instances Amazon mises en service pour votre plan de contrôle Kubernetes ne sont soumises à aucune obligation de placement matériel spécifique sur l'ensemble de la capacité sous-jacente disponible sur votre Outpost.

Vous pouvez utiliser des groupes de placement pour répondre aux exigences de haute disponibilité de votre cluster Amazon EKS local sur un Outpost. En spécifiant un groupe de placement lors de la création du cluster, vous influencez le placement des instances du plan de contrôle Kubernetes. Les instances sont réparties sur du matériel sous-jacent indépendant (racks ou hôtes), minimisant ainsi l'impact des instances corrélées en cas de panne matérielle.

Le type de répartition que vous pouvez configurer dépend du nombre de racks Outpost dont vous disposez dans votre déploiement.

- Déploiements avec un ou deux racks physiques dans un seul avant-poste logique : vous devez disposer d'au moins trois hôtes configurés avec le type d'instance que vous avez choisi pour vos

instances de plan de contrôle Kubernetes. Un groupe de placement de spread utilisant le spread au niveau de l'hôte garantit que toutes les instances du plan de contrôle Kubernetes s'exécutent sur des hôtes distincts au sein des racks sous-jacents disponibles dans votre déploiement Outpost.

- Déploiements comportant au moins trois racks physiques dans un seul avant-poste logique : au moins trois hôtes doivent être configurés avec le type d'instance que vous avez choisi pour vos instances de plan de contrôle Kubernetes. Un groupe de placement de spread utilisant le spread au niveau du rack garantit que toutes les instances du plan de contrôle Kubernetes s'exécutent sur des racks distincts dans votre déploiement Outpost. Vous pouvez également utiliser le groupe de placement par répartition au niveau des hôtes comme décrit dans l'option précédente.

La création du groupe de placement souhaité vous incombe. Vous spécifiez le groupe de placement lorsque vous appelez l'API `CreateCluster`. Pour plus d'informations sur les groupes de placement et sur la manière de les créer, consultez la section Groupes de placement dans le guide de EC2 l'utilisateur Amazon.

- Lorsqu'un groupe de placement est spécifié, de la capacité doit être disponible sur votre Outpost pour créer un cluster Amazon EKS local. La capacité varie selon que vous utilisez le type de répartition au niveau des hôtes ou des racks. Si la capacité est insuffisante, le cluster reste dans `Creating` cet état. Vous pouvez voir le message `Insufficient Capacity Error` dans le champ de santé de la réponse de l'API DescribeCluster. Vous devez libérer de la capacité pour que le processus de création puisse progresser.
- Lors des mises à jour de la plateforme et des versions du cluster local Amazon EKS, les instances du plan de contrôle Kubernetes de votre cluster sont remplacées par de nouvelles instances selon une stratégie de mise à jour continue. Au cours de ce processus de remplacement, chaque instance de plan de contrôle est arrêtée et son emplacement est libéré. Une nouvelle instance mise à jour est provisionnée à sa place. L'instance mise à jour peut être placée dans l'emplacement qui a été libéré. Si l'emplacement est occupé par une autre instance indépendante et que la capacité disponible est insuffisante pour répondre aux exigences topologiques de répartition, le cluster reste à l'état `Updating`. Vous pouvez voir le message `Insufficient Capacity Error` correspondant dans le champ de santé de la réponse de l'API DescribeCluster. Vous devez libérer de la capacité pour que le processus de mise à jour puisse progresser et rétablir les niveaux de haute disponibilité antérieurs.
- Vous pouvez créer un maximum de 500 groupes de placement par compte dans chaque AWS région. Pour plus d'informations, consultez la section Règles générales et limitations du guide de EC2 l'utilisateur Amazon.

Résoudre les problèmes liés aux clusters Amazon EKS locaux sur Outposts AWS

Cette rubrique traite de certaines erreurs courantes que vous pourriez rencontrer lorsque vous utilisez des clusters locaux, ainsi que des solutions. Les clusters locaux sont similaires aux clusters Amazon EKS dans le cloud, mais ils sont gérés différemment par Amazon EKS.

> **Important**
>
> Ne mettez jamais fin à une instance de `Kubernetes` plan de contrôle du cluster local EKS gérée exécutée sur Outpost, sauf instruction explicite du Support. AWS La mise hors service de ces instances représente un risque pour la disponibilité du service de cluster local, notamment la perte du cluster local en cas de résiliation simultanée de plusieurs instances. Les instances du `Kubernetes` plan de contrôle du cluster local EKS sont identifiées par la balise `eks-local:controlplane-name` sur la console d' EC2 instance.

Comportement de l'API

Les clusters locaux sont créés via l'API Amazon EKS, mais sont exécutés de manière asynchrone. Cela signifie que les demandes adressées à l'API Amazon EKS aboutissent immédiatement pour les clusters locaux. Toutefois, ces demandes peuvent aboutir, effectuer une interruption immédiate en raison d'erreurs de validation d'entrée, ou échouer et comporter des erreurs de validation descriptives. Ce comportement est similaire à celui de l'API Kubernetes.

Les clusters locaux ne passent pas à un `FAILED` statut. Amazon EKS tente de rapprocher l'état du cluster de l'état souhaité demandé par l'utilisateur de manière continue. Par conséquent, un cluster local peut rester dans l'état `CREATING` pendant une longue période, jusqu'à ce que le problème sous-jacent soit résolu.

Décrire le champ de santé du cluster

Les problèmes de cluster locaux peuvent être découverts à l'aide de la commande describe-cluster Amazon EKS AWS CLI. Les problèmes de clusters locaux sont mis en évidence par le `cluster.health` champ de réponse de la `describe-cluster` commande. Le message contenu dans ce champ inclut un code d'erreur, un message descriptif et une ressource associée IDs. Ces informations sont uniquement disponibles via l'API et la AWS CLI Amazon EKS. Dans l'exemple suivant, remplacez *`my-cluster`* par le nom de votre cluster local.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

L'exemple qui suit illustre un résultat.

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

Si le problème est irréparable, vous devrez peut-être supprimer le cluster local et en créer un nouveau. Par exemple, vous essayez de provisionner un cluster avec un type d'instance qui n'est pas disponible sur votre Outpost. Le tableau suivant répertorie les erreurs courantes liées à l'état.

Scénario d'erreur	Code	Message	ResourceIds
Les sous-réseaux fournis sont introuvables.	`ResourceNotFound`	`The subnet ID` *`subnet-id`* `does not exist`	Tous les sous-réseaux fournis IDs
Les sous-réseaux fournis n'appartiennent pas au même VPC.	`ConfigurationConflict`	`Subnets specified must belong to the same VPC`	Tous les sous-réseaux fournis IDs
Certains sous-réseaux fournis n'appartiennent pas à l'avant-poste spécifié.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `expected to be in` *`outpost-arn`* `, but is in` *`other-outpost-arn`*	ID de sous-réseau problématique

Scénario d'erreur	Code	Message	ResourceIds
Certains sous-réseaux fournis n'appartiennent à aucun Outpost.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `is not part of any Outpost`	ID de sous-réseau problématique
Certains sous-réseaux fournis ne disposent pas de suffisamment d'adresses libres pour créer des interfaces réseau élastiques pour les instances du plan de contrôle.	`ResourceLimitExceeded`	`The specified subnet does not have enough free addresses to satisfy the request.`	ID de sous-réseau problématique
Le type d'instance de plan de contrôle spécifié n'est pas pris en charge sur votre Outpost.	`ConfigurationConflict`	`The instance type` *`type`* `is not supported in Outpost` *`outpost-arn`*	ARN de cluster

Scénario d'erreur	Code	Message	ResourceIds
Vous avez mis fin à une EC2 instance Amazon du plan de contrôle ou vous avez `run-instance` réussi, mais l'état observé change de `Terminated`. Cela peut se produire pendant un certain temps après que votre Outpost se soit reconnecté et que des erreurs internes d'Amazon EBS aient entraîné l'échec d'un flux de travail EC2 interne d'Amazon.	`InternalFailure`	`EC2 instance state "Terminated" is unexpected`	ARN de cluster
Vous avez une capacité insuffisante sur votre Outpost. Cela peut également se produire lors de la création d'un cluster si un avant-poste est déconnecté de la AWS région.	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	ARN de cluster
Votre compte a dépassé votre quota de groupes de sécurité.	`ResourceLimitExceeded`	Message d'erreur renvoyé par l' EC2 API Amazon	ID du VPC cible

Scénario d'erreur	Code	Message	ResourceIds
Votre compte a dépassé votre quota d'interface réseau Elastic.	`ResourceLimitExceeded`	Message d'erreur renvoyé par l' EC2 API Amazon	l'ID du sous-réseau cible
Les instances du plan de contrôle n'étaient pas accessibles via AWS Systems Manager. Pour la résolution, consultez the section called "Les instances du plan de contrôle ne sont pas accessibles via AWS Systems Manager".	`ClusterUnreachable`	Les instances du plan de contrôle Amazon EKS ne sont pas accessibles via SSM. Vérifiez votre SSM et votre configuration réseau, et consultez la documentation de résolution des problèmes EKS sur les Outposts.	EC2 Instance Amazon IDs
Une erreur s'est produite lors de l'obtention de détails pour un groupe de sécurité géré ou une interface réseau Elastic.	Basé sur le code d'erreur du EC2 client Amazon.	Message d'erreur renvoyé par l' EC2 API Amazon	Tous les groupes de sécurité sont gérés IDs
Une erreur s'est produite lors de l'autorisation ou de la révocation des règles d'entrée du groupe de sécurité. Cela s'applique à la fois aux groupes de sécurité du cluster et du plan de contrôle.	Basé sur le code d'erreur du EC2 client Amazon.	Message d'erreur renvoyé par l' EC2 API Amazon	ID de groupe de sécurité problématique

Scénario d'erreur	Code	Message	ResourceIds
Une erreur s'est produite lors de la suppression d'une interface réseau Elastic d'une instance du plan de contrôle.	Basé sur le code d'erreur du EC2 client Amazon.	Message d'erreur renvoyé par l' EC2 API Amazon	ID d'interface réseau Elastoc problématique

Le tableau suivant répertorie les erreurs provenant d'autres AWS services présentées dans le champ de santé de la `describe-cluster` réponse.

Code EC2 d'erreur Amazon	Code de problème de santé du cluster	Description
`AuthFailure`	`AccessDenied`	Cette erreur peut se produire pour différentes raisons. La raison la plus courante est que vous avez accidentellement supprimé une balise que le service utilise pour réduire la portée de la politique de rôle lié à un service du plan de contrôle. Dans ce cas, Amazon EKS ne pourra plus gérer ni surveiller ces AWS ressources.
`UnauthorizedOperation`	`AccessDenied`	Cette erreur peut se produire pour différentes raisons. La raison la plus courante est que vous avez accidentellement supprimé une balise que le service utilise pour réduire la portée de la politique de rôle lié à un service du plan

Code EC2 d'erreur Amazon	Code de problème de santé du cluster	Description
		de contrôle. Dans ce cas, Amazon EKS ne pourra plus gérer ni surveiller ces AWS ressources.
`InvalidSubnetID.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque l'ID de sous-réseau correspondant aux règles d'entrée d'un groupe de sécurité est introuvable.
`InvalidPermission.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque les autorisations relatives aux règles d'entrée d'un groupe de sécurité ne sont pas correctes.
`InvalidGroup.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque le groupe des règles d'entrée d'un groupe de sécurité est introuvable.
`InvalidNetworkInterfaceID.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque l'ID d'interface réseau pour les règles d'entrée d'un groupe de sécurité est introuvable.
`InsufficientFreeAddressesInSubnet`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota de ressources du sous-réseau est dépassé.
`InsufficientCapacityOnOutpost`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota de capacité de l'avant-poste est dépassé.

Code EC2 d'erreur Amazon	Code de problème de santé du cluster	Description
`NetworkInterfaceLimitExceeded`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota de l'interface réseau Elastic est dépassé.
`SecurityGroupLimitExceeded`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota du groupe de sécurité est dépassé.
`VcpuLimitExceeded`	`ResourceLimitExceeded`	Cela est observé lors de la création d'une EC2 instance Amazon dans un nouveau compte. L'erreur peut être similaire à ce qui suit : "You `have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit` http://aws.amazon.com/contact-us/ec2-request `to request an adjustment to this limit."`
`InvalidParameterValue`	`ConfigurationConflict`	Amazon EC2 renvoie ce code d'erreur si le type d'instance spécifié n'est pas pris en charge sur l'Outpost.
Toutes les autres erreurs	`InternalFailure`	Aucune

Impossible de créer ou de modifier des clusters

Les clusters locaux nécessitent des autorisations et des politiques différentes de celles des clusters Amazon EKS qui sont hébergés dans le cloud. Lorsqu'un cluster ne parvient pas à se créer et génère une `InvalidPermissions` erreur, vérifiez que le rôle de cluster que vous utilisez est associé à la politique EKSLocal OutpostClusterPolicy gérée par Amazon. Tous les autres appels d'API nécessitent le même ensemble d'autorisations que les clusters Amazon EKS dans le cloud.

Le cluster est bloqué à l'état **`CREATING`**

Le temps nécessaire à la création d'un cluster local varie en fonction de plusieurs facteurs. Ces facteurs incluent la configuration de votre réseau, la configuration d'Outpost et la configuration du cluster. En général, un cluster local est créé et passe à l'état `ACTIVE` dans les 15 à 20 minutes. Si un cluster local reste dans l'état `CREATING`, vous pouvez appeler `describe-cluster` pour obtenir des informations sur la cause dans le champ de sortie `cluster.health`.

Les problèmes les plus courants sont les suivants :

- Votre cluster ne peut pas se connecter à l'instance du plan de contrôle depuis la AWS région dans laquelle se trouve Systems Manager. Vous pouvez le vérifier en appelant `aws ssm start-session --target` *`instance-id`* depuis un hôte bastion de la région. Si cette commande ne fonctionne pas, vérifiez si Systems Manager est en cours d'exécution sur l'instance du plan de contrôle. Une autre solution consiste à supprimer le cluster, puis à le recréer.
- Les instances du plan de contrôle ne sont pas créées en raison des autorisations clés KMS pour les volumes EBS. Avec les clés KMS gérées par l'utilisateur pour les volumes EBS chiffrés, les instances du plan de contrôle s'arrêteront si la clé n'est pas accessible. Si les instances sont mises hors service, passez à une clé KMS AWS gérée ou assurez-vous que votre politique de clé gérée par l'utilisateur accorde les autorisations nécessaires au rôle de cluster.
- Il se peut que les instances du plan de contrôle de Systems Manager n'aient pas accès à Internet. Vérifiez si le sous-réseau que vous avez fourni lors de la création du cluster possède une passerelle NAT et un VPC avec une passerelle Internet. Utilisez l'analyseur d'accessibilité VPC pour vérifier si l'instance du plan de contrôle peut atteindre la passerelle Internet. Pour plus d'informations, consultez Getting started with VPC Reachability Analyzer (langue française non garantie).
- Le rôle ARN que vous avez fourni ne contient pas de politiques. Vérifiez si la politique AWS gérée : Amazon EKSLocal OutpostClusterPolicy a été supprimée du rôle. Cela peut également se produire si une AWS CloudFormation pile est mal configurée.

- Tous les sous-réseaux fournis doivent être associés au même Outpost et pouvoir communiquer entre eux. Lorsque plusieurs sous-réseaux sont spécifiés lors de la création d'un cluster, Amazon EKS tente de répartir les instances du plan de contrôle sur plusieurs sous-réseaux.
- Les groupes de sécurité gérés par Amazon EKS sont appliqués sur l'interface réseau Elastic. Cependant, d'autres éléments de configuration tels que les règles de pare-feu NACL peuvent entrer en conflit avec les règles de l'interface réseau Elastic.

La configuration du VPC et du DNS du sous-réseau est mal configurée ou manquante

Consultez l'article Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur Outposts. AWS

Le cluster est bloqué à l'état **`UPDATING`**

Amazon EKS met automatiquement à jour tous les clusters locaux existants vers les dernières versions de plate-forme pour la version mineure de Kubernetes correspondante. Pour plus d'informations sur les versions de la plateforme, reportez-vous àthe section called "Versions de la plateforme EKS".

Lors du déploiement automatique d'une version de plate-forme, le statut du cluster passe à. `UPDATING` Le processus de mise à jour consiste à remplacer toutes les instances du plan de contrôle Kubernetes par de nouvelles instances contenant les derniers correctifs de sécurité et corrections de bogues publiés pour la version mineure correspondante de Kubernetes. En général, le processus de mise à jour d'une plate-forme de cluster locale se termine en moins de 30 minutes et le cluster revient à `ACTIVE` son état. Si un cluster local reste dans `UPDATING` cet état pendant une période prolongée, vous pouvez appeler `describe-cluster` pour vérifier les informations sur la cause dans le champ `cluster.health` de sortie.

Amazon EKS garantit qu'au moins 2 instances du plan de contrôle Kubernetes sur 3 sont des nœuds de cluster sains et opérationnels afin de maintenir la disponibilité du cluster local et d'éviter les interruptions de service. Si un cluster local est bloqué, c'est généralement parce qu'un problème d'infrastructure ou de configuration empêche de garantir la disponibilité minimale de deux instances au cas où le processus se poursuivrait. `UPDATING` Le processus de mise à jour cesse donc de progresser pour protéger l'interruption de service du cluster local.

Il est important de dépanner un cluster local bloqué et de remédier à `UPDATING` la cause première afin que le processus de mise à jour puisse se terminer et rétablir le cluster local `ACTIVE` avec la haute disponibilité de 3 instances du plan de contrôle Kubernetes.

Ne mettez fin à aucune `Kubernetes` instance de cluster local EKS gérée sur Outposts, sauf indication explicite du Support AWS . Cela est particulièrement important pour les clusters locaux bloqués, car il `UPDATING` est fort probable qu'un autre nœud du plan de contrôle ne soit pas complètement sain et que l'arrêt de la mauvaise instance puisse entraîner une interruption du service et une perte de données du cluster local.

Les problèmes les plus courants sont les suivants :

- Une ou plusieurs instances du plan de contrôle ne peuvent pas se connecter à System Manager en raison d'un changement de configuration réseau depuis la création du cluster local. Vous pouvez le vérifier en appelant `aws ssm start-session --target instance-id` depuis un hôte bastion de la région. Si cette commande ne fonctionne pas, vérifiez si Systems Manager est en cours d'exécution sur l'instance du plan de contrôle.
- Les nouvelles instances du plan de contrôle ne peuvent pas être créées en raison des autorisations clés KMS pour les volumes EBS. Avec les clés KMS gérées par l'utilisateur pour les volumes EBS chiffrés, les instances du plan de contrôle s'arrêteront si la clé n'est pas accessible. Si les instances sont mises hors service, passez à une clé KMS AWS gérée ou assurez-vous que votre politique de clé gérée par l'utilisateur accorde les autorisations nécessaires au rôle de cluster.
- Les instances du plan de contrôle de Systems Manager ont peut-être perdu l'accès à Internet. Vérifiez si le sous-réseau fourni lors de la création du cluster possède une passerelle NAT et un VPC avec une passerelle Internet. Utilisez l'analyseur d'accessibilité VPC pour vérifier si l'instance du plan de contrôle peut atteindre la passerelle Internet. Pour plus d'informations, consultez Getting started with VPC Reachability Analyzer (langue française non garantie). Si vos réseaux privés ne disposent pas de connexion Internet sortante, assurez-vous que tous les points de terminaison VPC et de passerelle requis sont toujours présents dans le sous-réseau régional de votre cluster (voir). the section called “Accès aux services par sous-réseau AWS”
- Le rôle ARN que vous avez fourni ne contient pas de politiques. Vérifiez si la politique AWS gérée : Amazon n'EKSLocalOutpostClusterPolicya pas été supprimée du rôle.
- L'une des nouvelles instances du plan de contrôle Kubernetes a peut-être connu un échec de démarrage inattendu. Veuillez déposer un ticket auprès du AWS Support Center pour obtenir des conseils supplémentaires sur le dépannage et la collecte des journaux dans ce cas exceptionnel.

Impossible de joindre des nœuds à un cluster

- Problèmes AMI :

  - Vous utilisez une AMI non prise en charge. Vous devez utiliser la version v20220620 ou une version ultérieure pour créer des nœuds avec Amazon Linux optimisé Amazon EKS optimisé pour AMIs Amazon Linux.
  - Si vous avez utilisé un AWS CloudFormation modèle pour créer vos nœuds, assurez-vous qu'il n'utilise pas une AMI non prise en charge.
- L'authentificateur AWS IAM est manquant `ConfigMap` : s'il est absent, vous devez le créer. Pour plus d'informations, consultez the section called "Appliquer la `ConfigMap aws-auth` à votre cluster".
- Le mauvais groupe de sécurité est utilisé – Assurez-vous d'utiliser `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid` pour le groupe de sécurité de vos composants master. Le groupe de sécurité sélectionné est modifié AWS CloudFormation pour autoriser un nouveau groupe de sécurité chaque fois que la pile est utilisée.
- Suite à des étapes inattendues liées à un VPC de lien privé – Des données CA incorrectes (`--b64-cluster-ca`) ou le mauvais point de terminaison de l'API (`--apiserver-endpoint`) sont transmis.

Collecte des journaux

Lorsqu'un avant-poste est déconnecté de la AWS région à laquelle il est associé, le cluster Kubernetes continuera probablement à fonctionner normalement. Toutefois, si le cluster ne fonctionne pas correctement, suivez les étapes de résolution des problèmes décrites dans Préparer les clusters Amazon EKS locaux sur AWS Outposts pour les déconnexions réseau. Si vous rencontrez d'autres problèmes, contactez le AWS Support. AWS Support peut vous aider à télécharger et à exécuter un outil de collecte de journaux. Ainsi, vous pouvez collecter des journaux à partir de vos instances de plan de contrôle du cluster Kubernetes et les envoyer au AWS support technique pour une enquête plus approfondie.

Les instances du plan de contrôle ne sont pas accessibles via AWS Systems Manager

Lorsque les instances du plan de contrôle Amazon EKS ne sont pas accessibles via AWS Systems Manager (Systems Manager), Amazon EKS affiche le message d'erreur suivant pour votre cluster.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

Pour résoudre ce problème, assurez-vous que votre VPC et vos sous-réseaux répondent aux exigences de la section Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur AWS Outposts et que vous avez suivi les étapes décrites dans Configuration du gestionnaire de session dans le guide de l'utilisateur de Systems Manager. AWS

Créez des nœuds Amazon Linux sur AWS Outposts

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Amazon Linux sur un Outpost qui s'enregistrent auprès de votre cluster Amazon EKS. Le cluster peut se trouver dans le AWS cloud ou dans un avant-poste.

- Un Outpost existant. Pour plus d'informations, consultez What is AWS Outposts.
- Un cluster Amazon EKS existant. Pour déployer un cluster sur le AWS cloud, consultezthe section called “Créer un cluster ”. Pour déployer un cluster sur un Outpost, consultez the section called “Exécuter des clusters locaux”.
- Supposons que vous créiez vos nœuds dans un cluster sur le AWS Cloud et que vous disposiez de sous-réseaux dans la AWS région où AWS Outposts, AWS Wavelength ou Local AWS Zones sont activés. Alors, ces sous-réseaux ne doivent pas avoir été transmis lors de la création de votre cluster. Si vous créez vos nœuds dans un cluster sur un Outpost, vous devez avoir transmis un sous-réseau Outpost lors de la création de votre cluster.
- (Recommandé pour les clusters sur le AWS cloud) Le module complémentaire Amazon VPC CNI pour Kubernetes est configuré avec son propre rôle IAM auquel est attachée la politique IAM nécessaire. Pour de plus amples informations, veuillez consulter the section called “Configuration pour IRSA”. Les clusters locaux ne prennent pas en charge les rôles IAM pour les comptes de service.

Vous pouvez créer un groupe de nœuds Amazon Linux autogéré avec `eksctl` ou AWS Management Console (avec un AWS CloudFormation modèle). Vous pouvez également utiliser Terraform.

Vous pouvez créer un groupe de nœuds autogéré pour un cluster local à l'aide des outils suivants décrits dans cette page :

- the section called “`eksctl`”
- the section called “AWS Management Console”

Important

- Le groupe de nœuds autogéré inclut EC2 les instances Amazon dans votre compte. Ces instances ne sont pas automatiquement mises à niveau lorsque vous ou Amazon EKS mettez à jour la version du plan de contrôle en votre nom. Aucun groupe de nœuds autogéré n'indique dans la console qu'il doit être mis à jour. Vous pouvez afficher la version `kubelet` installée sur un nœud en sélectionnant le nœud dans la boîte de dialogue Nœuds dans la Présentation de votre cluster pour déterminer quels nœuds doivent être mis à jour. Vous devez mettre à jour les nœuds manuellement. Pour de plus amples informations, veuillez consulter the section called "Méthodes de mise à jour".
- Les certificats utilisés par kubelet sur vos nœuds autogérés sont émis avec une expiration d'un an. Par défaut, la rotation des certificats n'est pas activée (voir : https://kubernetes).io/docs/reference/config-api/kubelet-config.v1beta1/# kubelet-config-k 8 s-io-v 1beta1-KubeletConfiguration), cela signifie que si vous avez un nœud autogéré qui fonctionne depuis plus d'un an, il ne pourra plus s'authentifier auprès de l'API Kubernetes.
- À titre de bonne pratique, nous recommandons aux clients de mettre régulièrement à jour leurs groupes de nœuds autogérés afin de recevoir CVEs les correctifs de sécurité de la dernière AMI optimisée pour Amazon EKS. La mise à jour de l'AMI utilisée dans les groupes de nœuds autogérés déclenche également la recréation des nœuds et garantit qu'ils ne rencontrent aucun problème en raison de l'expiration des certificats Kubelet.
- Vous pouvez également activer la rotation des certificats clients (voir : https://kubernetes).io/docs/tasks/tls/certificate-rotation/) lors de la création des groupes de nœuds autogérés pour s'assurer que les certificats kubelet sont renouvelés lorsque le certificat actuel approche de son expiration.

`eksctl`

Pour lancer des nœuds Linux autogérés à l'aide de **`eksctl`**

1. Installez la version `0.210.0` ou une version ultérieure de l'outil de ligne de `eksctl` commande installé sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Si votre cluster se trouve AWS dans le cloud et que la politique IAM gérée par Amazoneks_CNI_Policy est attachée au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service

Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA". Si votre cluster se trouve sur votre Outpost, la politique doit être associée à votre rôle de nœud.

3. La commande suivante crée un groupe de nœuds dans un cluster existant. Le cluster doit avoir été créé à l'aide d'`eksctl`. Remplacer *`al-nodes`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas comporter plus de 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Si votre cluster existe sur un Outpost, remplacez *`id`* par l'ID d'un sous-réseau Outpost. Si votre cluster existe dans le AWS cloud, remplacez-le *`id`* par l'ID d'un sous-réseau que vous n'avez pas spécifié lors de la création de votre cluster. Remplacez *`instance-type`* par un type d'instance pris en charge par votre Outpost. Remplacez les valeurs de *`example values`* restantes par vos propres valeurs. Par défaut, les nœuds sont créés avec la même version de Kubernetes que le plan de contrôle.

   Remplacez *`instance-type`* par un type d'instance disponible sur votre Outpost.

   *`my-key`*Remplacez-le par le nom de votre paire de EC2 clés Amazon ou de votre clé publique. Cette clé est utilisée pour SSH dans vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

   Créez votre groupe de nœuds avec la commande suivante.

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-type instance-
type \
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2 --
subnet-ids subnet-id
```

   Si votre cluster est déployé sur le AWS cloud :

   - Le groupe de nœuds que vous déployez peut attribuer des `IPv4` adresses aux pods à partir d'un bloc CIDR différent de celui de l'instance. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée".

- Le groupe de nœuds que vous déployez ne nécessite pas d'accès Internet sortant. Pour de plus amples informations, veuillez consulter the section called "Clusters privés".

Pour une liste complète de toutes les options et valeurs par défaut disponibles, consultez AWS Outposts Support `eksctl` dans la documentation.

- Si les nœuds ne parviennent pas à rejoindre le cluster, consultez the section called "Les nœuds ne parviennent pas à joindre le cluster" les sections Résolution des problèmes liés aux clusters et nœuds Amazon EKS et Résoudre les problèmes liés the section called "Impossible de joindre des nœuds à un cluster" aux clusters Amazon EKS locaux sur AWS Outposts.
- L'exemple qui suit illustre un résultat. Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.

AWS Management Console

Étape 1 : Lancez des nœuds Linux autogérés à l'aide de AWS Management Console

1. Téléchargez la dernière version du AWS CloudFormation modèle.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. Ouvrez la AWS CloudFormation console.
3. Choisissez Create stack (Créer une pile), puis sélectionnez Avec de nouvelles ressources (standard).
4. Pour Spécifier un modèle, sélectionnez Upload a template file (Télécharger un fichier de modèle), puis sélectionnez Choose file (Choisir un fichier). Sélectionnez le fichier `amazon-eks-nodegroup.yaml` que vous avez téléchargé à une étape précédente, puis sélectionnez Next (Suivant).
5. Dans la page Specify stack details (Spécifier les détails de la pile), saisissez les paramètres suivants, puis choisissez Next (Suivant) :
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *`al-nodes`*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère

alphanumérique et ne doit pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.

- ClusterName: Entrez le nom de votre cluster. Si ce nom ne correspond pas au nom de votre cluster, vos nœuds ne peuvent pas rejoindre le cluster.
- ClusterControlPlaneSecurityGroup: Choisissez la SecurityGroupsvaleur dans la AWS CloudFormation sortie que vous avez générée lors de la création de votre VPC.

  Les étapes suivantes montrent une opération permettant de récupérer le groupe applicable.

  i. Ouvrez la console Amazon EKS.

  ii. Choisissez le nom du cluster.

  iii. Choisissez l'onglet Networking (Mise en réseau).

  iv. Utilisez la valeur Groupes de sécurité supplémentaires comme référence lorsque vous effectuez une sélection ClusterControlPlaneSecurityGroupdans la liste déroulante.
- NodeGroupName: Entrez le nom de votre groupe de nœuds. Ce nom pourra être utilisé ultérieurement pour identifier le groupe de nœuds Auto Scaling créé pour vos nœuds.
- NodeAutoScalingGroupMinSize: Entrez le nombre minimum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeAutoScalingGroupDesiredCapacity: Entrez le nombre de nœuds que vous souhaitez atteindre lors de la création de votre pile.
- NodeAutoScalingGroupMaxSize: Entrez le nombre maximum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeInstanceType: Choisissez un type d'instance pour vos nœuds. Si votre cluster s'exécute sur le AWS Cloud, pour plus d'informations, consultezthe section called "Types d' EC2 instances Amazon". Si votre cluster s'exécute sur un Outpost, vous ne pouvez sélectionner qu'un type d'instance disponible sur votre Outpost.
- NodeImageIdSSMParam: prérempli avec le paramètre Amazon EC2 Systems Manager d'une AMI récemment optimisée pour Amazon EKS pour une version variable de Kubernetes. Pour utiliser une autre version mineure de Kubernetes prise en charge par Amazon EKS, *1.XX* remplacez-la par un autre fichier eks/latest/userguide/kubernetes -versions.html [version prise en charge, type="documentation "]. Nous vous recommandons de spécifier la même version de Kubernetes que celle de votre cluster.

  Pour utiliser une AMI accélérée optimisée pour Amazon EKS, remplacez-la *amazon-linux-2* paramazon-linux-2-gpu. Pour utiliser une AMI Arm optimisée pour Amazon EKS, *amazon-linux-2* remplacez-la paramazon-linux-2-arm64.

> **Note**
>
> Le nœud Amazon EKS AMIs est basé sur Amazon Linux. Vous pouvez suivre les événements liés à la sécurité ou à la confidentialité d'Amazon Linux dans le centre de sécurité Amazon Linux en choisissant l'onglet correspondant à la version de votre choix. Vous pouvez également vous abonner au flux RSS applicable. Les événements de sécurité et de confidentialité incluent une présentation du problème, les packages concernés et la manière de mettre à jour vos instances pour résoudre le problème.

- NodeImageId: (Facultatif) Si vous utilisez votre propre AMI personnalisée (au lieu d'une AMI optimisée pour Amazon EKS), entrez un ID d'AMI de nœud pour votre AWS région. Si vous spécifiez une valeur ici, elle remplace toutes les valeurs du NodeImageIdSSMParamchamp.
- NodeVolumeSize: Spécifiez une taille de volume racine pour vos nœuds, en GiB.
- NodeVolumeType: Spécifiez un type de volume racine pour vos nœuds.
- KeyName: Entrez le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

> **Note**
>
> Si vous ne fournissez pas de paire de clés ici, la création de la AWS CloudFormation pile échoue.

- BootstrapArguments: Il existe plusieurs arguments facultatifs que vous pouvez transmettre à vos nœuds. Pour de plus amples informations, veuillez consulter Utilisation du script d'amorçage sur GitHub. Si vous ajoutez des nœuds à un cluster local Amazon EKS sur des AWS Outposts (où les instances du plan de contrôle Kubernetes s'exécutent sur des AWS Outposts) et que le cluster ne dispose pas de connexion Internet d'entrée et de sortie (également appelée clusters privés), vous devez fournir les arguments bootstrap suivants (sur une seule ligne).

```
--b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://${APISERVER_ENDPOINT}
 --enable-local-outpost true --cluster-id ${CLUSTER_ID}
```

Pour récupérer les valeurs pour `CLUSTER_CAAPISERVER_ENDPOINT`, et `CLUSTER_ID` de votre cluster local Amazon EKS, exécutez les commandes AWS CLI suivantes. Remplacez cluster-name par le nom de votre cluster et de votre région (par exemple, us-east-1) par la région de votre cluster. AWS

```
echo "CLUSTER_CA=$(aws eks describe-cluster --name cluster-name --region region --
query cluster.certificateAuthority.data --output text)"

echo "APISERVER_ENDPOINT=$(aws eks describe-cluster --name cluster-name --region
 region --query cluster.endpoint --output text)"

echo "CLUSTER_ID=$(aws eks describe-cluster --name cluster-name --region region --
query cluster.id --output text)"
```

- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Vous pouvez le désactiver IMDSv1. Pour empêcher l'utilisation des futurs nœuds et pods du groupe de nœuds IMDSv1, définissez Disable IMDSv1 sur true. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance. Pour plus d'informations sur la façon d'en restreindre l'accès sur vos nœuds, consultez Restreindre l'accès au profil d'instance affecté au composant master.
- VpcId: Entrez l'ID du VPC que vous avez créé. Avant de choisir un VPC, passez en revue les exigences et les considérations relatives au VPC.
- Subnets : si votre cluster se trouve sur un Outpost, choisissez au moins un sous-réseau privé dans votre VPC. Avant de choisir les sous-réseaux, consultez Exigences et considérations requises pour les sous-réseaux. Vous pouvez consulter les sous-réseaux privés en ouvrant le lien de chaque sous-réseau depuis l'onglet Networking (Mise en réseau) de votre cluster.

6. Sélectionnez les choix que vous souhaitez sur la page Configure stack options (Configurer les options de la pile), puis choisissez Next (Suivant).
7. Cochez la case à gauche de J'accuse réception AWS CloudFormation susceptible de créer des ressources IAM. , puis choisissez Create stack.
8. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
9. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin lors de la configuration de vos nœuds pour Amazon EKS.

Étape 2 : Permettre aux nœuds de rejoindre votre cluster

1. Vérifiez si vous avez déjà appliqué le `ConfigMap` `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si vous voyez un `ConfigMap` `aws-auth`, mettez-le à jour si nécessaire.

   a. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez une nouvelle entrée `mapRoles` si nécessaire. Définissez la `rolearn` valeur sur la NodeInstanceRolevaleur que vous avez enregistrée lors de la procédure précédente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. Enregistrez le fichier et quittez votre éditeur de texte.

3. Si vous avez reçu un message d'erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le stock `ConfigMap`.

   a. Téléchargez la mappe de configuration.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

   b. Dans le `aws-auth-cm.yaml` fichier, définissez la `rolearn` NodeInstanceRolevaleur que vous avez enregistrée lors de la procédure précédente. Pour ce faire, utilisez un éditeur de texte ou remplacez *`my-node-instance-role`* et exécutez la commande suivante :

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

   c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

Saisissez `Ctrl+C` pour revenir à une invite de shell.

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

Si les nœuds ne parviennent pas à rejoindre le cluster, consultez the section called "Les nœuds ne parviennent pas à joindre le cluster" les sections Résolution des problèmes liés aux clusters et nœuds Amazon EKS et Résoudre les problèmes liés the section called "Impossible de joindre des nœuds à un cluster" aux clusters Amazon EKS locaux sur AWS Outposts.

5. Installez le pilote CSI Amazon EBS. Pour plus d'informations, consultez la section Installation sur GitHub. Dans la section Set up driver permission (Configurer les autorisations du pilote), assurez-vous de suivre les instructions de l'option Using IAM instance profile (Utilisation du profil d'instance IAM). Vous devez utiliser la classe de stockage `gp2`. La classe `gp3` de stockage n'est pas prise en charge.

   Pour créer une classe de stockage `gp2` sur votre cluster, suivez les étapes suivantes.

   a. Exécutez la commande ci-dessous pour créer un fichier`gp2-storage-class.yaml`.

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
```

```
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Nœuds GPU uniquement) Si vous avez choisi un type d'instance GPU et une AMI accélérée optimisée pour Amazon EKS, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes DaemonSet sur votre cluster. *vX.X.X*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Étape 3 : actions supplémentaires

1. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.
2. Si votre cluster est déployé sur un Outpost, ignorez cette étape. Si votre cluster est déployé sur le AWS Cloud, les informations suivantes sont facultatives. Si la politique IAM gérée par Amazoneks_CNI_Policy est attachée au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

Présentation de l'intelligence artificielle (IA) et du Machine Learning (ML) sur Amazon EKS

Amazon Elastic Kubernetes Service (EKS) est une plateforme Kubernetes gérée qui permet aux entreprises de déployer, gérer et dimensionner les charges de travail liées à l'IA et au machine learning (ML) avec une flexibilité et un contrôle inégalés. Construit sur l'écosystème open source Kubernetes, EKS vous permet de tirer parti de votre expertise Kubernetes existante, tout en s'intégrant parfaitement aux outils et services open source. AWS

Que vous entraîniez des modèles à grande échelle, que vous exécutiez des inférences en ligne en temps réel ou que vous déployiez des applications d'IA génératives, EKS fournit les performances, l'évolutivité et la rentabilité qu'exigent vos AI/ML projets.

Pourquoi choisir EKS pour l'IA et le ML ?

EKS est une plateforme Kubernetes gérée qui vous aide à déployer et à gérer des charges de travail complexes. AI/ML Construit sur l'écosystème open source Kubernetes, il s'intègre AWS aux services, fournissant le contrôle et l'évolutivité nécessaires aux projets avancés. Pour les équipes qui découvrent AI/ML les déploiements, les compétences Kubernetes existantes sont transférées directement, ce qui permet une orchestration efficace de plusieurs charges de travail.

EKS prend en charge tout, de la personnalisation du système d'exploitation à la mise à l'échelle du calcul, et sa base open source favorise la flexibilité technologique, préservant ainsi le choix pour les décisions futures en matière d'infrastructure. La plate-forme fournit les performances et les options de réglage requises par les AI/ML charges de travail, en prenant en charge des fonctionnalités telles que :

- Contrôle complet du cluster pour affiner les coûts et les configurations sans abstractions cachées
- Latence inférieure à la seconde pour les charges de travail d'inférence en temps réel en production
- Personnalisations avancées telles que le multi-instance GPUs, les stratégies multi-cloud et le réglage au niveau du système d'exploitation
- Possibilité de centraliser les charges de travail à l'aide d'EKS en tant qu'orchestrateur unifié entre les pipelines AI/ML

Principaux cas d'utilisation

Amazon EKS fournit une plate-forme robuste pour un large éventail de AI/ML charges de travail, prenant en charge diverses technologies et modèles de déploiement :

- Inférence en temps réel (en ligne) : EKS fournit des prévisions immédiates sur les données entrantes, telles que la détection des fraudes, avec une latence inférieure à une seconde à l'aide d'outils tels que TorchServeTriton Inference Server et sur les instances KServeAmazon EC2 Inf1 et Inf2. Ces charges de travail bénéficient d'une mise à l'échelle dynamique avec Karpenter et KEDA, tout en tirant parti d'Amazon EFS pour le découpage des modèles entre les pods. Amazon ECR Pull Through Cache (PTC) accélère les mises à jour des modèles, et les volumes de données Bottlerocket associés aux volumes optimisés pour Amazon EBS garantissent un accès rapide aux données.
- Formation générale sur les modèles : les organisations utilisent EKS pour former des modèles complexes sur de grands ensembles de données sur de longues périodes à l'aide de Kubeflow Training Operator (KRO), de Ray Serve et de Torch Distributed Elastic sur des instances Amazon P4d EC2 et Amazon Trn1. EC2 Ces charges de travail sont prises en charge par la planification par lots avec des outils tels que Volcano, Yunikorn et Kueue. Amazon EFS permet de partager les points de contrôle des modèles, tandis qu'Amazon S3 gère les politiques relatives au cycle de vie import/export des modèles pour la gestion des versions.
- Pipelines de génération augmentée (RAG) : EKS gère les chatbots de support client et les applications similaires en intégrant des processus de récupération et de génération. Ces charges de travail utilisent souvent des outils tels qu'Argo Workflows et Kubeflow pour l'orchestration, des bases de données vectorielles comme Pinecone, Weaviate ou Amazon OpenSearch, et exposent les applications aux utilisateurs via l'Application Load Balancer Controller (LBC). NVIDIA NIM optimise l'utilisation du GPU, tandis que Prometheus et Grafana surveillent l'utilisation des ressources.
- Déploiement de modèles d'IA générative : les entreprises déploient des services de création de contenu en temps réel sur EKS, tels que la génération de texte ou d'images, à l'aide de Ray Serve, vLLM et Triton Inference Server sur Amazon EC2 G5 et les accélérateurs Inferentia. Ces déploiements optimisent les performances et l'utilisation de la mémoire pour les modèles à grande échelle. JupyterHubpermet le développement itératif, Gradius fournit des interfaces Web simples et le pilote S3 Mountpoint CSI permet de monter des compartiments S3 en tant que systèmes de fichiers pour accéder à des fichiers de modèles volumineux.
- Inférence par lots (hors ligne) : les organisations traitent efficacement de grands ensembles de données grâce à des tâches planifiées avec AWS Batch ou Volcano. Ces charges de travail

utilisent souvent des instances Inf1 et Inf2 pour les AWS puces Inferentia, des EC2 instances Amazon EC2 G4dn pour NVIDIA T4 ou des instances de processeur c5 et c6i GPUs, maximisant ainsi l'utilisation des ressources pendant les heures creuses pour les tâches d'analyse. Le SDK AWS Neuron et les pilotes GPU NVIDIA optimisent les performances, tout en MIG/TS permettant le partage du GPU. Les solutions de stockage incluent Amazon S3, Amazon EFS et FSx pour Lustre, avec des pilotes CSI pour différentes classes de stockage. La gestion des modèles s'appuie sur des outils tels que Kubeflow Pipelines, Argo Workflows et Ray Cluster, tandis que la surveillance est assurée par Prometheus, Grafana et des outils de surveillance de modèles personnalisés.

Études de cas

Les clients choisissent Amazon EKS pour diverses raisons, telles que l'optimisation de l'utilisation du GPU ou l'exécution de charges de travail d'inférence en temps réel avec une latence inférieure à une seconde, comme le montrent les études de cas suivantes. Pour obtenir la liste de toutes les études de cas relatives à Amazon EKS, consultez AWS Customer Success Stories.

- Unitary traite 26 millions de vidéos par jour en utilisant l'IA pour modérer le contenu, ce qui nécessite une inférence à haut débit et à faible latence. Elle a réduit de 80 % les temps de démarrage des conteneurs, garantissant ainsi une réponse rapide aux événements de dimensionnement liés aux fluctuations du trafic.
- Miro, la plateforme de collaboration visuelle supportant 70 millions d'utilisateurs dans le monde, a enregistré une réduction de 80 % des coûts de calcul par rapport à ses précédents clusters Kubernetes autogérés.
- Synthesia, qui propose un service de création vidéo générative basé sur l'intelligence artificielle permettant aux clients de créer des vidéos réalistes à partir d'instructions textuelles, a multiplié par 30 le débit de formation des modèles ML.
- Harri, fournisseur de technologies RH pour le secteur de l'hôtellerie, a accéléré de 90 % sa mise à l'échelle en réponse aux pics de demande et a réduit ses coûts de calcul de 30 % en migrant vers AWS les processeurs Graviton.
- Ada Support, une société d'automatisation du service client basée sur l'IA, a réussi à réduire de 15 % ses coûts de calcul tout en augmentant de 30 % son efficacité informatique.
- Snorkel AI, qui permet aux entreprises de créer et d'adapter des modèles de base et de grands modèles linguistiques, a réalisé des économies de coûts de plus de 40 % en mettant en œuvre des mécanismes de dimensionnement intelligents pour leurs ressources GPU.

Commencez à utiliser le Machine Learning sur EKS

Pour commencer à planifier et à utiliser les plateformes et les charges de travail de Machine Learning sur EKS AWS dans le cloud, passez à la the section called "Ressources" section.

Exécution de charges de travail d'inférence en ligne en temps réel sur Amazon EKS

Cette section est conçue pour vous aider à déployer et à exploiter des charges de travail d'inférence en ligne en temps réel sur Amazon Elastic Kubernetes Service (EKS). Vous y trouverez des conseils sur la création de clusters optimisés avec des nœuds accélérés par GPU, l'intégration de AWS services de stockage et de mise à l'échelle automatique, le déploiement d'exemples de modèles à des fins de validation, ainsi que des considérations architecturales clés telles que le découplage des tâches du processeur et du processeur graphique, la sélection des types d'instance appropriés AMIs et la garantie d'une exposition à faible latence des points de terminaison d'inférence.

Rubriques

- Guide de configuration de clusters basé sur les meilleures pratiques pour l'inférence en temps réel sur Amazon EKS

Guide de configuration de clusters basé sur les meilleures pratiques pour l'inférence en temps réel sur Amazon EKS

Introduction

Ce guide propose une procédure pratique pour configurer un cluster Amazon Elastic Kubernetes Service (EKS) optimisé pour les charges de travail d'inférence en ligne en temps réel, en intégrant les meilleures pratiques élaborées par des experts du début à la fin. AWS Il utilise une architecture EKS Quickstart bien ancrée, c'est-à-dire un ensemble de pilotes, de types d'instances et de configurations sélectionnés conformément aux AWS meilleures pratiques en matière de modèles, d'accélérateurs et de scalabilité. Cette approche vous permet d'éviter de sélectionner les paramètres du cluster, ce qui vous permet de mettre rapidement en place un cluster fonctionnel et préconfiguré. En cours de route, nous déploierons des exemples de charges de travail pour valider votre configuration, expliquer les concepts architecturaux clés (tels que le découplage des tâches liées au processeur des calculs gourmands en GPU), répondre aux questions courantes (par exemple, pourquoi choisir Bottlerocket

AMI plutôt que 023 ?) AL2 , et décrivez les prochaines étapes pour étendre les capacités de votre cluster.

Conçu spécifiquement pour les ingénieurs du Machine Learning (ML) et de l'intelligence artificielle (IA), les administrateurs de plateformes, les opérateurs et les data/AI spécialistes qui découvrent l'écosystème EKS pour la AWS première fois, ce guide suppose une connaissance de Kubernetes mais aucune expérience préalable d'EKS. Il est conçu pour vous aider à comprendre les étapes et les processus nécessaires pour que les charges de travail d'inférence en ligne en temps réel soient opérationnelles. Ce guide explique les éléments essentiels de la création d'un cluster d'inférence à nœud unique, notamment le provisionnement des ressources GPU, l'intégration du stockage pour les artefacts du modèle, la sécurisation de l'accès aux AWS services et la mise en évidence des points de terminaison d'inférence. Dans l'ensemble, il met l'accent sur une conception résiliente et à faible latence pour les applications destinées aux utilisateurs, telles que la détection des fraudes, les chatbots en temps réel et l'analyse des sentiments dans les systèmes de feedback client.

Dans ce guide, nous nous concentrons exclusivement sur la mise en place d'un point de départ prescriptif fondamental à l'aide d'instances G5. EC2 Si vous recherchez des configurations de clusters ou des end-to-end flux de travail AWS spécifiques à Inferentia, consultez nos the section called “Configurer des clusters d'inférence avec Inferentia” ateliers dans. the section called “Ressources”

Avant de commencer

Avant de commencer, assurez-vous d'avoir effectué les tâches suivantes :

- Configuration de votre environnement pour Amazon EKS
- Installez la dernière version d'eksctl
- Installez Helm
- (Facultatif) Installez Docker
- (Facultatif) Installez la CLI NGC

Architecture

L'inférence en ligne en temps réel fait référence au processus d'utilisation d'un modèle d'apprentissage automatique entraîné pour générer des prédictions ou des sorties sur les flux de données entrants avec une latence minimale. Par exemple, il permet de détecter les fraudes en temps réel, de classer des images ou de générer des graphiques de connaissances en réponse

aux entrées des utilisateurs. L'architecture d'un système d'inférence en ligne en temps réel fournit des prédictions d'apprentissage automatique à faible latence dans les applications destinées aux utilisateurs en dissociant la gestion du trafic Web lié au processeur des calculs d'intelligence artificielle gourmands en GPU. Ce processus s'inscrit généralement dans un écosystème d'applications plus vaste et inclut souvent des services de backend, de frontend, de vecteur et de modèle, en mettant l'accent sur des composants spécialisés pour permettre une mise à l'échelle indépendante, un développement parallèle et une résilience en cas de défaillance. L'isolation des tâches d'inférence sur du matériel GPU dédié et l'exploitation d'interfaces telles que APIs et WebSockets garantissent une simultanéité élevée, un traitement rapide de modèles tels que les transformateurs et des interactions avec les utilisateurs via le frontend. Notez que bien que les bases de données vectorielles et les pipelines RAG (Retrieval Augmented Generation) jouent souvent un rôle important dans les systèmes d'inférence en temps réel, ces composants ne sont pas abordés dans ce guide. Au minimum, une architecture typique comprend souvent :

- Service frontal : Sert d'interface utilisateur, gère la logique côté client, affiche le contenu dynamique et facilite les interactions en temps réel. Il communique avec le service principal pour lancer des demandes d'inférence et afficher les résultats, en adressant souvent des demandes au service principal qui l'utilise WebSockets pour les mises à jour en continu ou pour l'échange de données structurées. APIs Ce service ne nécessite généralement pas d'équilibreur de charge dédié, car il peut être hébergé sur des réseaux de diffusion de contenu (CDNs), comme AWS CloudFront pour les actifs statiques, ou servi directement à partir de serveurs Web, le dimensionnement étant géré via des groupes d'auto-dimensionnement si nécessaire pour le contenu dynamique.
- Service principal : agit en tant qu'orchestrateur de l'application, gérant la logique métier telle que l'authentification des utilisateurs, la validation des données et la coordination des services (par exemple, via APIs pour les RESTful points de terminaison ou WebSockets pour les connexions persistantes). Il communique avec le service d'inférence, s'adapte indépendamment sur le multicœur CPUs et la RAM pour gérer le trafic Web élevé sans compter sur celui-ci GPUs, et nécessite souvent un équilibreur de charge (tel que Application Load AWS Balancer ou Network Load Balancer) pour répartir les demandes entrantes sur plusieurs instances, en particulier dans les scénarios de concurrence élevée. Un contrôleur d'entrée peut également gérer les règles d'accès et de routage externes pour améliorer la sécurité et la régulation du trafic.
- Service d'inférence : sert de base aux calculs d'intelligence artificielle, fonctionnant GPUs avec suffisamment de VRAM (par exemple, 8 à 12 Go pour des modèles tels que Distilbert) pour effectuer des intégrations vectorielles, l'extraction de connaissances et l'inférence de modèles (par exemple, exposée pour les demandes par lots ou APIs pour le streaming en temps réel) à l'aide de modèles personnalisés ou open source WebSockets . Cette isolation empêche les conflits

de dépendance, permet de mettre à jour les modèles sans interruption de service et permet une mise à l'échelle horizontale avec équilibrage de charge pour plusieurs demandes simultanées. Pour exposer efficacement le modèle de service, celui-ci est généralement installé derrière un équilibreur de charge chargé de répartir les charges de travail liées au GPU entre les instances répliquées, tandis qu'une ressource ou un contrôleur d'entrée (tel qu'un contrôleur d'entrée ALB intégré AWS) gère le routage externe, la terminaison SSL et le transfert basé sur le chemin afin de garantir un accès sécurisé et efficace sans surcharger le personnel. GPUs

Présentation de la solution

Les systèmes d'inférence en ligne en temps réel nécessitent une architecture résiliente et performante capable de fournir une latence extrêmement faible tout en gérant des pics de trafic imprévisibles et importants. Cette présentation de la solution explique comment les AWS composants suivants fonctionnent ensemble dans le cluster Amazon EKS que nous allons créer pour garantir que notre cluster est capable d'héberger et de gérer des modèles d'apprentissage automatique qui fournissent des prévisions immédiates sur les données en temps réel avec un délai minimal pour les utilisateurs finaux.

- Instances Amazon G5 — Pour les tâches d'inférence gourmandes en GPU, nous utilisons les types d' EC2 instances G5 g5.xlarge et g5.2xlarge, qui comportent un (1) seul GPU NVIDIA A10G avec 24 Go de mémoire (EC2 par exemple, 8 milliards de paramètres chez). FP16 Basés sur l'architecture NVIDIA Ampere, GPUs ils sont alimentés par des processeurs NVIDIA A10G Tensor Core GPUs et AMD EPYC de 2e génération, prennent en charge une bande passante réseau de 4 à 8 VCPUs, jusqu'à 10 Gbit/s et 250 à 450 Go de stockage NVMe SSD local, garantissant un transfert rapide des données et une puissance de calcul pour les modèles complexes, ce qui les rend idéaux pour les tâches d'inférence à faible latence et à haut débit. Le choix d'un type d' EC2 instance est spécifique à l'application, dépend de votre modèle (par exemple, modèle d'image, de vidéo, de texte) et de vos exigences en matière de latence et de débit. Par exemple, si vous utilisez un modèle d'image et/ou de vidéo, vous souhaiterez peut-être utiliser des EC2 instances P5 pour une latence optimale en temps réel. Nous vous recommandons de commencer par les EC2 instances G5, car elles constituent un bon point de départ pour une mise en service rapide, puis d'évaluer si elles sont adaptées à vos charges de travail grâce à des tests de performance. Pour les cas plus avancés, pensez aux EC2 instances G6.
- Instances Amazon EC2 m7g : pour les tâches gourmandes en ressources processeur telles que le prétraitement des données, le traitement des demandes d'API, l'hébergement du contrôleur Karpenter, les modules complémentaires et d'autres composants du système, nous utilisons le type d'instance m5.xlarge m7g. EC2 Les instances m7G sont des instances ARM dotées de 4 VCPUs,

de 16 Go de mémoire, d'une bande passante réseau allant jusqu'à 12,5 Gbit/s et alimentées par des processeurs Graviton3. AWS Le choix d'un type d' EC2 instance est spécifique à l'application et dépend des exigences de calcul, de mémoire et d'évolutivité de votre charge de travail. Pour les charges de travail optimisées pour le calcul, vous pouvez envisager les EC2 instances C7g, qui utilisent également des processeurs Graviton3 mais sont optimisées pour des performances de calcul supérieures à celles des instances M7g dans certains cas d'utilisation. Par ailleurs, les nouvelles EC2 instances C8g (lorsqu'elles sont disponibles) offrent des performances de calcul jusqu'à 30 % supérieures à celles des instances C7g. Nous vous recommandons de commencer par les EC2 instances m7G pour leur rentabilité et leur compatibilité avec un large éventail de charges de travail (par exemple, les serveurs d'applications, les microservices, les serveurs de jeu, les magasins de données de taille moyenne), puis d'évaluer si elles conviennent à vos charges de travail grâce à des tests de performance.

- Pilote CSI Mountpoint Amazon S3 — Pour les charges de travail sur des instances à GPU unique où plusieurs pods partagent un GPU (par exemple, plusieurs pods programmés sur le même nœud pour utiliser ses ressources GPU), nous utilisons le pilote CSI Mountpoint S3 pour optimiser l'utilisation de la mémoire, ce qui est essentiel pour des tâches telles que l'inférence de grands modèles dans des configurations sensibles aux coûts et peu complexes. Il présente les buckets Amazon S3 comme un système de fichiers de type POSIX disponible pour le cluster Kubernetes, qui permet aux modules d'inférence de lire les artefacts des modèles (par exemple, les poids des modèles) directement en mémoire sans avoir à les télécharger au préalable, et de saisir des ensembles de données à l'aide d'opérations de fichiers standard. De plus, S3 dispose d'une capacité de stockage pratiquement illimitée et accélère les charges de travail d'inférence gourmandes en données. Le choix d'un pilote CSI de stockage est spécifique à l'application et dépend des exigences de débit et de latence de votre charge de travail. Bien que le pilote CSI FSx pour OpenZFS offre une latence inférieure à la milliseconde pour les volumes persistants partagés aléatoires I/O ou entièrement compatibles POSIX entre les nœuds, nous vous recommandons de commencer par le pilote CSI Mountpoint S3 en raison de son évolutivité, de ses coûts réduits pour les grands ensembles de données et de son intégration au stockage d'objets géré par S3 pour les modèles d'inférence lourds (par exemple, les entrées du modèle de streaming), puis d'évaluer s'il convient à vos charges de travail grâce à des tests de performance.
- Agent d'identité EKS Pod — Pour permettre l'accès aux AWS services, nous utilisons l'agent EKS Pod Identity, qui utilise un seul principal de service et facilite les associations de rôles IAM au niveau du pod au sein du cluster Amazon EKS. EKS Pod Identity offre une alternative rationalisée à l'approche traditionnelle des rôles IAM pour les comptes de service (IRSA) en utilisant un seul principal de service (pods.eks.amazonaws.com) au lieu de s'appuyer sur des fournisseurs OIDC individuels pour chaque cluster, ce qui facilite l'attribution des autorisations. En outre, il permet de

réutiliser les rôles sur plusieurs clusters et prend en charge des fonctionnalités avancées telles que les balises de session de rôle IAM et les rôles Target IAM.

- Agent de surveillance des nœuds EKS — Pour garantir la disponibilité et la fiabilité continues des services d'inférence, nous utilisons l'agent de surveillance des nœuds EKS avec réparation automatique, qui détecte et remplace automatiquement les nœuds défectueux, minimisant ainsi les temps d'arrêt. Il surveille en permanence les nœuds pour détecter les problèmes liés au matériel, au noyau, au réseau et au stockage à l'aide de contrôles de santé améliorés (par exemple KernelReady,, NetworkingReady). Pour les nœuds GPU, il détecte les défaillances spécifiques à l'accélérateur, initie une correction progressive en bouclant les nœuds défectueux, en attendant 10 minutes que les problèmes de GPU transitoires soient résolus et en remplaçant les nœuds au bout de 30 minutes en cas de défaillance persistante.
- AMI Bottlerocket — Pour renforcer la sécurité de notre cluster EKS, nous utilisons l'AMI Bottlerocket, qui inclut uniquement les composants essentiels nécessaires au fonctionnement des conteneurs et offre des temps de démarrage minimaux pour une mise à l'échelle rapide. Le choix d'une AMI de nœud est spécifique à l'application et dépend des exigences de personnalisation, de sécurité et d'évolutivité de votre charge de travail. Bien que l'AMI AL2 023 offre une plus grande flexibilité pour les installations et les personnalisations au niveau de l'hôte (par exemple, en spécifiant un répertoire de cache dédié dans une configuration PV/PVC sans nœuds supplémentaires), nous vous recommandons de commencer par l'AMI Bottlerocket pour son encombrement réduit et son optimisation intégrée pour les charges de travail conteneurisées (par exemple, microservices, serveurs d'inférence, évolutifs APIs), puis d'évaluer si elle convient à vos charges de travail grâce à des tests de performance.
- AWS Load Balancer Controller (LBC) — Pour exposer les points de terminaison d'inférence en temps réel, nous utilisons le Load Balancer Controller, qui approvisionne et gère automatiquement les équilibreurs de AWS charge d'application () pour le trafic et les équilibreurs de charge réseau (ALBs) pour le HTTP/HTTPS trafic en fonction des ressources Kubernetes Ingress et Service, permettant ainsi l'intégration de modèles d'inférence avec des clients externes. NLBs TCP/UDP En outre, il prend en charge des fonctionnalités telles que le routage basé sur les chemins pour distribuer les demandes d'inférence sur plusieurs pods ou nœuds, garantissant ainsi l'évolutivité lors des pics de trafic et minimisant la latence grâce à des optimisations AWS natives telles que le multiplexage des connexions et les bilans de santé.

1. Créez votre cluster EKS

Dans cette étape, nous créons un cluster avec des nœuds CPU et un groupe de nœuds gérés à l'aide d'un modèle eksctl AWS CloudFormation alimenté par un modèle eksctl ClusterConfig.

L'initialisation du cluster avec uniquement des nœuds CPU nous permet d'utiliser Karpenter exclusivement pour gérer les nœuds gourmands en CPU et en GPU afin d'optimiser l'allocation des ressources à l'aide de Karpenter NodePools , que nous créerons ultérieurement. Pour prendre en charge nos charges de travail d'inférence en temps réel, nous fournissons au cluster l'AMI EKS Bottlerocket, l'agent de surveillance des nœuds EKS, l'agent EKS Pod Identity, le pilote Mountpoint S3 CSI, le contrôleur Load AWS Balancer (LBC) et les pilotes kube-proxy, vpc-cni et coredns. Les instances m7g.xlarge seront utilisées pour les tâches du système CPU, y compris l'hébergement du contrôleur Karpenter, des modules complémentaires et d'autres composants du système.

Par défaut, `eksctl` créera un VPC dédié pour le cluster avec un bloc CIDR de. `192.168.0.0/16` Le VPC comprend trois sous-réseaux publics et trois sous-réseaux privés, chacun étant réparti sur trois zones de disponibilité différentes (ou deux AZs dans la `us-east-1` région), ce qui constitue la méthode idéale pour déployer les charges de travail Kubernetes. Le modèle déploie également une passerelle Internet, fournissant un accès Internet aux sous-réseaux publics via des routes par défaut dans leurs tables de routage et une passerelle NAT unique dans l'un des sous-réseaux publics, les routes par défaut dans les tables de routage des sous-réseaux privés dirigeant le trafic sortant via la passerelle NAT pour l'accès à Internet. Pour en savoir plus sur cette configuration, consultez la section Déployer des nœuds sur des sous-réseaux privés.

Vérifiez vos informations d'identification

Vérifiez si vos informations d'identification AWS CLI sont valides et si vous pouvez vous authentifier auprès des AWS services :

```
aws sts get-caller-identity
```

En cas de succès, la CLI renverra des informations sur votre AWS identité (UserId, votre compte et votre ARN).

Vérifier la disponibilité de l'instance

Les types d'instances G5 ne sont pas disponibles dans toutes les régions. Vérifiez la région la plus proche. Par exemple :

```
aws ec2 describe-instance-types --instance-types g5.xlarge g5.2xlarge --region us-
east-1
```

En cas de succès, le type d'instance G5 est disponible dans la région que vous avez spécifiée.

L'AMI Bottlerocket n'est pas disponible dans toutes les régions. Vérifiez en récupérant un ID AMI Bottlerocket pour la région la plus proche. Par exemple :

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.33/arm64/latest/
image_id \
    --region us-east-1 --query "Parameter.Value" --output text
```

En cas de succès, l'AMI Bottlerocket est disponible dans la région que vous avez spécifiée.

Préparez votre environnement

Définissez d'abord les variables d'environnement suivantes dans une nouvelle fenêtre de terminal. Remarque : veillez à remplacer les exemples d'espaces réservés par vos valeurs uniques, notamment le nom du cluster, la région souhaitée, la version de Karpenter et la version de Kubernetes.

> **Tip**
>
> Certaines variables (telles que `${AWS_REGION}` et`${K8S_VERSION}`) sont définies au début du bloc, puis référencées dans des commandes ultérieures pour des raisons de cohérence et pour éviter les répétitions. Assurez-vous d'exécuter les commandes dans l'ordre afin que ces valeurs soient correctement exportées et puissent être utilisées dans les définitions suivantes.

```
export TEMPOUT="$(mktemp)"
export K8S_VERSION=1.33
export KARPENTER_VERSION="1.5.0"
export AWS_REGION="us-east-1"
export EKS_CLUSTER_NAME="eks-rt-inference-${AWS_REGION}"
export S3_BUCKET_NAME="eks-rt-inference-models-${AWS_REGION}-$(date +%s)"
export NVIDIA_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}-nvidia/x86_64/latest/image_id --query
 Parameter.Value --output text)"
export STANDARD_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}/arm64/latest/image_id --query Parameter.Value --
output text)"
export AWS_ACCOUNT_ID="$(aws sts get-caller-identity --query Account --output text)"
export ALIAS_VERSION="$(aws ssm get-parameter --name "/aws/service/eks/optimized-
ami/${K8S_VERSION}/amazon-linux-2023/x86_64/standard/recommended/image_id" --query
 Parameter.Value | xargs aws ec2 describe-images --query 'Images[0].Name' --image-ids |
 sed -r 's/^.*(v[[:digit:]]+).*$/\1/')"
```

Création des rôles et des politiques requis

Karpenter a besoin de rôles et de politiques IAM spécifiques (par exemple, le rôle IAM du contrôleur Karpenter, le profil d'instance et les politiques) pour gérer les EC2 instances en tant que nœuds de travail Kubernetes. Il utilise ces rôles pour effectuer des actions telles que le lancement et la résiliation d' EC2 instances, le balisage des ressources et l'interaction avec d'autres AWS services. Créez les rôles et les politiques de Karpenter à l'aide du fichier cloudformation.yaml de Karpenter :

```
curl -fsSL https://raw.githubusercontent.com/aws/karpenter-provider-aws/v
${KARPENTER_VERSION}/website/content/en/preview/getting-started/getting-started-with-
karpenter/cloudformation.yaml > "${TEMPOUT}" \
&& aws cloudformation deploy \
  --stack-name "Karpenter-${EKS_CLUSTER_NAME}" \
  --template-file "${TEMPOUT}" \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides "ClusterName=${EKS_CLUSTER_NAME}"
```

Le AWS LBC a besoin d'une autorisation pour approvisionner et gérer des équilibreurs de AWS charge, par exemple ALBs pour créer des ressources Ingress ou NLBs pour des services de type similaire. `LoadBalancer` Nous définirons cette politique d'autorisation lors de la création du cluster. Lors de la création du cluster, nous créerons le compte de service avec eksctl dans le. ClusterConfig Créez la politique LBC IAM :

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document "$(curl -fsSL https://raw.githubusercontent.com/kubernetes-sigs/
aws-load-balancer-controller/v2.13.0/docs/install/iam_policy.json)"
```

Lorsque le pilote CSI Mountpoint S3 est installé, ses DaemonSet modules sont configurés pour utiliser un compte de service pour l'exécution. Le pilote CSI Mountpoint for Mountpoint S3 doit être autorisé pour interagir avec le compartiment Amazon S3 que vous créez plus loin dans ce guide. Nous définirons cette politique d'autorisation lors de la création du cluster. Lors de la création du cluster, nous créerons le compte de service avec eksctl dans le. ClusterConfig Créez la politique IAM S3 :

```
aws iam create-policy \
    --policy-name S3CSIDriverPolicy \
    --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:PutObject\", \"s3:AbortMultipartUpload
```

```
\", \"s3:DeleteObject\", \"s3:ListBucket\"], \"Resource\": [\"arn:aws:s3:::
${S3_BUCKET_NAME}\", \"arn:aws:s3:::${S3_BUCKET_NAME}/*\"]}]}"
```

Remarque : si un rôle portant ce nom existe déjà, donnez-lui un autre nom. Le rôle que nous créons au cours de cette étape est spécifique à votre cluster et à votre compartiment S3.

Créer le cluster

Dans ce modèle, eksctl crée automatiquement un compte de service Kubernetes pour EKS Pod Identity, Node Monitoring Agent, CoreDNS, Kubeproxy, le plug-in VPC CNI. À ce jour, le pilote CSI Mountpoint S3 n'est pas disponible pour EKS Pod Identity, nous avons donc créé un compte IRSA (IAM Roles for Service Account) et un point de terminaison OIDC. En outre, nous créons un compte de service pour le AWS Load Balancer Controller (LBC). Pour accéder aux nœuds Bottlerocket, eksctl attache automatiquement SSMManaged InstanceCore Amazon pour Bottlerocket afin de permettre des sessions shell sécurisées via SSM.

Dans le même terminal où vous définissez vos variables d'environnement, exécutez le bloc de commande suivant pour créer le cluster :

```
eksctl create cluster -f - <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ${EKS_CLUSTER_NAME}
  region: ${AWS_REGION}
  version: "${K8S_VERSION}"
  tags:
    karpenter.sh/discovery: ${EKS_CLUSTER_NAME}
    # Add more tags if needed for billing
iam:
  # Creates an OIDC endpoint and IRSA service account for the Mountpoint S3 CSI Driver
  # Uses the S3 CSI Driver policy for permissions
  withOIDC: true
  podIdentityAssociations:
  # Creates the pod identity association and service account
  # Uses the Karpenter controller IAM policy for permissions
  - namespace: "kube-system"
    serviceAccountName: karpenter
    roleName: ${EKS_CLUSTER_NAME}-karpenter
    permissionPolicyARNs:
```

```
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/KarpenterControllerPolicy-
${EKS_CLUSTER_NAME}
  # Creates the pod identity association and service account
  # Uses the AWS LBC policy for permissions
  - namespace: kube-system
    serviceAccountName: aws-load-balancer-controller
    createServiceAccount: true
    roleName: AmazonEKSLoadBalancerControllerRole
    permissionPolicyARNs:
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/AWSLoadBalancerControllerIAMPolicy
iamIdentityMappings:
- arn: "arn:aws:iam::${AWS_ACCOUNT_ID}:role/KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  username: system:node:{{EC2PrivateDNSName}}
  groups:
  - system:bootstrappers
  - system:nodes
managedNodeGroups:
  # Creates 2 CPU nodes for lightweight system tasks
  - name: ${EKS_CLUSTER_NAME}-m7-cpu
    instanceType: m7g.xlarge
    amiFamily: Bottlerocket
    desiredCapacity: 2
    minSize: 1
    maxSize: 10
    labels:
      role: cpu-worker
# Enable automatic Pod Identity associations for VPC CNI Driver, coreDNS, kube-proxy
addonsConfig:
  autoApplyPodIdentityAssociations: true
addons:
  # Installs the S3 CSI Driver addon and creates IAM role
  # Uses the S3 CSI Driver policy for IRSA permissions
  - name: aws-mountpoint-s3-csi-driver
    attachPolicyARNs:
      - "arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3CSIDriverPolicy"
  - name: eks-pod-identity-agent
  - name: eks-node-monitoring-agent
  - name: coredns
  - name: kube-proxy
  - name: vpc-cni
EOF
```

Ce processus prend plusieurs minutes. Si vous souhaitez surveiller l'état, consultez la AWS CloudFormationconsole.

2. Vérifiez l'état du nœud et du pod du cluster

Effectuons quelques vérifications de santé pour nous assurer que le cluster est prêt. Lorsque la commande précédente est terminée, visualisez les types d'instances et vérifiez que les nœuds de votre système CPU ont atteint l'`Ready`état requis à l'aide de la commande suivante :

```
kubectl get nodes -L node.kubernetes.io/instance-type
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                              STATUS   ROLES    AGE     VERSION
 INSTANCE-TYPE
ip-192-168-35-103.ec2.internal   Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
ip-192-168-7-15.ec2.internal     Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
```

Vérifiez toutes les associations Pod Identity et la manière dont elles associent un rôle à un compte de service dans un espace de noms du cluster à l'aide de la commande suivante :

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

La sortie doit indiquer les rôles IAM de Karpenter (« karpenter ») et du AWS LBC (» «). aws-load-balancer-controller

Vérifiez qu' DaemonSets ils sont disponibles :

```
kubectl get daemonsets -n kube-system
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                          DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR
          AGE
aws-node                      3       3       3     3          3         <none>
          12m
dcgm-server                   0       0       0     0          0
 kubernetes.io/os=linux 12m
```

```
eks-node-monitoring-agent      3       3       3     3          3
 kubernetes.io/os=linux 12m
eks-pod-identity-agent         3       3       3     3          3         <none>
          12m
kube-proxy                     3       3       3     3          3         <none>
          12m
s3-csi-node                    2       2       2     2          2
 kubernetes.io/os=linux 12m
```

Vérifiez que tous les addons sont installés sur le cluster :

```
eksctl get addons --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                           VERSION              STATUS    ISSUES    IAMROLE
                                      UPDATE AVAILABLE    CONFIGURATION VALUES    POD
 IDENTITY ASSOCIATION ROLES
aws-mountpoint-s3-csi-driver   v1.15.0-eksbuild.1   ACTIVE    0
 arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-us-east-1-addon-aws-m-Role1-
RAUjk4sJnc0L
coredns                        v1.12.1-eksbuild.2   ACTIVE    0
eks-node-monitoring-agent      v1.3.0-eksbuild.2    ACTIVE    0
eks-pod-identity-agent         v1.3.7-eksbuild.2    ACTIVE    0
kube-proxy                     v1.33.0-eksbuild.2   ACTIVE    0
metrics-server                 v0.7.2-eksbuild.3    ACTIVE    0
vpc-cni                        v1.19.5-eksbuild.1   ACTIVE    0
```

3. Installez Karpenter

Installez le contrôleur Karpenter sur les nœuds de travail de votre processeur (`cpu-worker`) afin d'optimiser les coûts et de préserver les ressources du GPU. Nous allons l'installer dans l'espace de noms « kube-system » et spécifier le compte de service « karpenter » que nous avons défini lors de la création du cluster. En outre, cette commande configure le nom du cluster et une file d'attente d'interruption d'instance Spot pour les nœuds du processeur. Karpenter utilisera l'IRSA pour assumer ce rôle IAM.

```
# Logout of helm registry before pulling from public ECR
helm registry logout public.ecr.aws

# Install Karpenter
```

```
helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version
 "${KARPENTER_VERSION}" --namespace "kube-system" --create-namespace  \
  --set "settings.clusterName=${EKS_CLUSTER_NAME}" \
  --set "settings.interruptionQueue=${EKS_CLUSTER_NAME}" \
  --set controller.resources.requests.cpu=1 \
  --set controller.resources.requests.memory=1Gi \
  --set controller.resources.limits.cpu=1 \
  --set controller.resources.limits.memory=1Gi \
  --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::
${AWS_ACCOUNT_ID}:role/${EKS_CLUSTER_NAME}-karpenter" \
  --wait
```

Le résultat attendu devrait ressembler à ceci :

```
Release "karpenter" does not exist. Installing it now.
Pulled: public.ecr.aws/karpenter/karpenter:1.5.0
Digest: sha256:9a155c7831fbff070669e58500f68d7ccdcf3f7c808dcb4c21d3885aa20c0a1c
NAME: karpenter
LAST DEPLOYED: Thu Jun 19 09:57:06 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Vérifiez que Karpenter est en cours d'exécution :

```
kubectl get pods -n kube-system -l app.kubernetes.io/name=karpenter
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                       READY   STATUS    RESTARTS   AGE
karpenter-555895dc-865bc   1/1     Running   0          5m58s
karpenter-555895dc-j7tk9   1/1     Running   0          5m58s
```

4. Configuration : Karpenter NodePools

Dans cette étape, nous configurons le processeur et le GPU Karpenter NodePools qui s'excluent mutuellement. Le `limits` champ de la NodePool spécification limite le maximum de ressources totales (par exemple, processeur, mémoire GPUs) que chacun NodePool peut consommer sur tous les nœuds provisionnés, empêchant ainsi le provisionnement de nœuds supplémentaires si ces limites sont dépassées. Tout en NodePools prenant en charge de larges catégories d'instances

(par exemplec,,g), la spécification de types d'instances spécifiques, de types de capacité et de limites de ressources vous permet d'estimer plus facilement les coûts de vos charges de travail à la demande. Dans ces derniers NodePools, nous utilisons un ensemble diversifié de types d'instances au sein de la famille d'instances G5. Cela permet à Karpenter de sélectionner automatiquement le type d'instance le plus approprié en fonction des demandes de ressources du pod, optimisant ainsi l'utilisation des ressources tout en respectant les limites totales NodePool du pod. Pour en savoir plus, consultez la section Création NodePools.

Configuration du GPU NodePool

Dans ce cadre NodePool, nous définissons des limites de ressources pour gérer le provisionnement des nœuds dotés de capacités GPU. Ces limites sont conçues pour plafonner le total des ressources sur tous les nœuds du pool, en autorisant jusqu'à 10 instances au total. Chaque instance peut être g5.xlarge (4 v, CPUs 16 GiB de mémoire, 1 GPU) ou g5.2xlarge (8 v, 32 GiB de mémoireCPUs, 1 GPU), à condition que le total de CPUs v ne dépasse pas 80, que la mémoire totale ne dépasse pas 320 Go et que le total ne dépasse pas 10. GPUs Par exemple, le pool peut provisionner 10 instances g5.2xlarge (80 v, CPUs 320 GiB, 10 GPUs) ou 10 instances g5.xlarge (40 v, CPUs 160 GiB, 10), ou une combinaison telle que 5 g5.xlarge et 5 g5.2xlarge (60 v, 240 GiB, 10 GPUs), garantissant ainsi une flexibilité basée sur les demandes de charge de travail tout en respectant les CPUs contraintes de ressources. GPUs

De plus, nous indiquons l'ID de la variante Nvidia de l'AMI Bottlerocket. Enfin, nous avons défini une politique d'interruption pour supprimer les nœuds vides au bout de 30 minutes (`consolidateAfter: 30m`) et fixé une durée de vie maximale des nœuds de 30 jours (`expireAfter: 720h`) afin d'optimiser les coûts et de préserver l'intégrité des nœuds pour les tâches gourmandes en GPU. Pour en savoir plus, consultez Désactiver la consolidation Karpenter pour les charges de travail sensibles aux interruptions et Utiliser ttlSecondsAfter Finished pour nettoyer automatiquement les tâches Kubernetes.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-a10g-inference-g5
spec:
  template:
    metadata:
      labels:
        role: gpu-worker
        gpu-type: nvidia-a10g
```

```
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["g5.xlarge", "g5.2xlarge"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
      nodeClassRef:
        name: gpu-a10g-inference-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "80"
    memory: "320Gi"
    nvidia.com/gpu: "10"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 30m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-a10g-inference-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${NVIDIA_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  tags:
    nvidia.com/gpu: "true"
EOF
```

Le résultat attendu devrait ressembler à ceci :

```
nodepool.karpenter.sh/gpu-a10g-inference-g5 created
ec2nodeclass.karpenter.k8s.aws/gpu-a10g-inference-ec2 created
```

Vérifiez que le NodePool est créé et qu'il est sain :

```
kubectl get nodepool gpu-a10g-inference-g5 -o yaml
```

Cherchez `status.conditions` comme `ValidationSucceeded: True` `NodeClassReady: True`, et confirmez `Ready: True` qu' NodePool il est sain.

Configuration du processeur NodePool

Dans ce cadre NodePool, nous avons défini des limites pour prendre en charge environ 50 instances, en fonction d'une charge de travail modérée du processeur (par exemple, 100 à 200 pods) et de quotas de AWS vCPU classiques (par exemple, 128 à 1152). Les limites sont calculées en supposant que l'échelle NodePool devrait atteindre 50 instances m7.xlarge : processeur (4 v CPUs par instance × 50 instances = 200 vCPUs) et mémoire (16 GiB par instance × 50 instances = 800 GiB). Ces limites sont conçues pour plafonner les ressources totales sur tous les nœuds du pool, en autorisant jusqu'à 50 instances de 7 g x large (chacune avec 4 V et CPUs 16 Go de mémoire), à condition que le total de v ne dépasse pas 200 et que la mémoire totale CPUs ne dépasse pas 800 Go.

De plus, nous indiquons l'ID de la variante standard de l'AMI Bottlerocket. Enfin, nous avons défini une politique d'interruption pour supprimer les nœuds vides au bout de 60 minutes (`consolidateAfter: 60m`) et fixé une durée de vie maximale des nœuds de 30 jours (`expireAfter: 720h`) afin d'optimiser les coûts et de préserver l'intégrité des nœuds pour les tâches gourmandes en GPU. Pour en savoir plus, consultez Désactiver la consolidation Karpenter pour les charges de travail sensibles aux interruptions et Utiliser ttlSecondsAfter Finished pour nettoyer automatiquement les tâches Kubernetes.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: cpu-inference-m7gxlarge
spec:
  template:
    metadata:
      labels:
```

```
        role: cpu-worker
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["m7g.xlarge"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
      taints:
        - key: role
          value: cpu-intensive
          effect: NoSchedule
      nodeClassRef:
        name: cpu-inference-m7gxlarge-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "200"
    memory: "800Gi"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 60m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: cpu-inference-m7gxlarge-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${STANDARD_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
EOF
```

Le résultat attendu devrait ressembler à ceci :

```
nodepool.karpenter.sh/cpu-inference-m7gxlarge created
ec2nodeclass.karpenter.k8s.aws/cpu-inference-m7gxlarge-ec2 created
```

Vérifiez que le NodePool est créé et qu'il est sain :

```
kubectl get nodepool cpu-inference-m7gxlarge -o yaml
```

Cherchez `status.conditions` comme `ValidationSucceeded: True` `NodeClassReady: True`, et confirmez `Ready: True` qu' NodePool il est sain.

5. Déployer un module GPU pour exposer un GPU

Vous avez besoin du Nvidia Device Plugin pour permettre à Kubernetes d'exposer les périphériques GPU au cluster Kubernetes. En général, vous devez déployer le plug-in en tant que tel DaemonSet ; toutefois, l'AMI Bottlerocket préinstalle le plug-in dans le cadre de l'AMI. Cela signifie que lorsque vous utilisez Bottlerocket AMIs, il n'est pas nécessaire de déployer le plugin pour appareil Nvidia. DaemonSet Pour en savoir plus, consultez la section consacrée au plugin Kubernetes Device. GPUs

Déployer un module d'échantillonnage

Karpenter agit de manière dynamique : il approvisionne les nœuds GPU lorsqu'une charge de travail (pod) demande des ressources GPU. Pour vérifier que les pods peuvent demander et utiliser GPUs, déployez un pod qui demande la `nvidia.com/gpu` ressource dans ses limites (par exemple,`nvidia.com/gpu: 1`). Pour en savoir plus sur ces étiquettes, voir Planifier les charges de travail en fonction des exigences en matière de GPU à l'aide d'étiquettes bien connues.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-nvidia-smi
spec:
  restartPolicy: OnFailure
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  nodeSelector:
    role: gpu-worker  # Matches GPU NodePool's label
  containers:
```

```
  - name: cuda-container
    image: nvidia/cuda:12.9.1-base-ubuntu20.04
    command: ["nvidia-smi"]
    resources:
      limits:
        nvidia.com/gpu: 1
      requests:
        nvidia.com/gpu: 1
EOF
```

Le résultat attendu devrait ressembler à ceci :

```
pod/gpu-ndivia-smi created
```

Attendez une minute, puis vérifiez si le Pod a le statut « En attente »ContainerCreating, « En cours », puis « Terminé » :

```
kubectl get pod gpu-nvidia-smi -w
```

Vérifiez que le nœud du pod appartient au GPU NodePool :

```
kubectl get node $(kubectl get pod gpu-nvidia-smi -o jsonpath='{.spec.nodeName}') -o
 custom-columns="Name:.metadata.name,Nodepool:.metadata.labels.karpenter\.sh/nodepool"
```

Le résultat attendu devrait ressembler à ceci :

```
Name                              Nodepool
ip-192-168-83-245.ec2.internal   gpu-a10g-inference-g5
```

Consultez les journaux du pod :

```
kubectl logs gpu-nvidia-smi
```

Le résultat attendu devrait ressembler à ceci :

```
Thu Jul 17 04:31:33 2025
+-----------------------------------------------------------------------------------------
+
```

```
| NVIDIA-SMI 570.148.08                Driver Version: 570.148.08         CUDA
 Version: 12.9 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA A10G                    On  | 00000000:00:1E.0 Off |                    0
 |
|  0%   30C    P8               9W / 300W |      0MiB / 23028MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------------------------------------------------------
+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
|  GPU        GI     CI        PID   Type   Process name                  GPU Memory
 |
|                     ID        ID                                           Usage
 |
|
=========================================================================================|
|  No running processes found
 |
+-----------------------------------------------------------------------------------------
+
```

6. (Facultatif) Préparer et télécharger les artefacts du modèle pour le déploiement

Au cours de cette étape, vous allez déployer un service de modélisation pour la classification des images en temps réel, en commençant par le téléchargement des pondérations des modèles dans un compartiment Amazon S3. À des fins de démonstration, nous utilisons le modèle open source de vision GPUNet-0 de NVIDIA GPUNet, qui prend en charge l'inférence à faible latence sur les images à l'aide de NVIDIA et de TensorRT. GPUs Ce modèle est préentraîné ImageNet, nous permet

de classer des objets sur des photos ou des flux vidéo à la volée et est considéré comme un petit modèle avec 11,9 millions de paramètres.

Configuration de votre environnement

Pour télécharger les poids de modèle GPUNet -0 Au cours de cette étape, vous devez accéder au catalogue NGC de NVIDIA et à Docker installé sur votre machine locale. Suivez ces étapes pour créer un compte gratuit et configurer la CLI NGC :

- Ouvrez un compte NGC gratuit et générez une clé API depuis le tableau de bord NGC (icône utilisateur > Configuration > Générer une clé API > Générer une clé personnelle > Catalogue NGC).
- Téléchargez et installez la CLI NGC (Linux/macOS/Windows) et configurez la CLI en utilisant :`ngc config set`. Entrez votre clé d'API lorsque vous y êtes invité ; définissez org sur `nvidia` et appuyez sur Entrée pour accepter les valeurs par défaut pour les autres. En cas de succès, vous devriez voir quelque chose comme : `Successfully saved NGC configuration to /Users/your-username/.ngc/config`

Vérifier les autorisations du compte de service

Avant de commencer, vérifiez les autorisations du compte de service Kubernetes :

```
kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o yaml
```

Lors de la création du cluster, nous avons associé la CSIDriver politique S3 à un rôle IAM et annoté le compte de service (« s3- csi-driver-sa »). Les pods du pilote CSI Mountpoint S3 héritent des autorisations du rôle IAM lorsqu'ils interagissent avec S3. Le résultat attendu devrait ressembler à ceci :

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-
us-east-1-addon-aws-m-Role1-fpXXjRYdKN8r
  creationTimestamp: "2025-07-17T03:55:29Z"
  labels:
    app.kubernetes.io/component: csi-driver
    app.kubernetes.io/instance: aws-mountpoint-s3-csi-driver
    app.kubernetes.io/managed-by: EKS
```

```
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: s3-csi-driver-sa
  namespace: kube-system
  resourceVersion: "2278"
  uid: 50b36272-6716-4c68-bdc3-c4054df1177c
```

Ajouter une tolérance

Le pilote S3 CSI s'exécute en tant que tel DaemonSet sur tous les nœuds. Les pods utilisent le pilote CSI sur ces nœuds pour monter les volumes S3. Pour lui permettre de se programmer sur nos nœuds GPU présentant des anomalies, ajoutez une tolérance aux éléments suivants : DaemonSet

```
kubectl patch daemonset s3-csi-node -n kube-system --type='json' -p='[{"op": "add",
 "path": "/spec/template/spec/tolerations/-", "value": {"key": "nvidia.com/gpu",
 "operator": "Exists", "effect": "NoSchedule"}}]'
```

Le résultat attendu devrait ressembler à ceci :

```
daemonset.apps/s3-csi-node patched
```

Charger les poids du modèle dans S3

Au cours de cette étape, vous allez créer un compartiment Amazon S3, télécharger les poids de modèle GPUNet -0 depuis NVIDIA GPU Cloud (NGC) et les charger dans le compartiment. Ces poids seront accessibles par notre application au moment de l'exécution à des fins d'inférence.

Créez votre compartiment Amazon S3 :

```
aws s3 mb s3://${S3_BUCKET_NAME} --region ${AWS_REGION}
```

Activez le contrôle de version S3 pour le compartiment, afin d'éviter que les suppressions et remplacements accidentels n'entraînent une perte de données immédiate et permanente :

```
aws s3api put-bucket-versioning --bucket ${S3_BUCKET_NAME} --versioning-configuration
 Status=Enabled
```

Appliquez une règle de cycle de vie au compartiment afin de supprimer les versions d'objets remplacées ou supprimées 14 jours après leur date d'expiration, de supprimer les marqueurs de suppression expirés et de supprimer les téléchargements en plusieurs parties incomplets au bout de 7 jours. Pour en savoir plus, consultez Exemples de configurations du cycle de vie S3.

```
aws s3api put-bucket-lifecycle-configuration --bucket $S3_BUCKET_NAME --lifecycle-
configuration '{"Rules":[{"ID":"LifecycleRule","Status":"Enabled","Filter":
{},"Expiration":{"ExpiredObjectDeleteMarker":true},"NoncurrentVersionExpiration":
{"NoncurrentDays":14},"AbortIncompleteMultipartUpload":{"DaysAfterInitiation":7}}]}'
```

Téléchargez les poids du modèle GPUNet -0 sur NGC. Par exemple, sur macOS :

```
ngc registry model download-version nvidia/dle/gpunet_0_pyt_ckpt:21.12.0_amp --dest ~/
downloads
```

> **Note**
>
> Vous devrez peut-être adapter cette commande de téléchargement à votre système d'exploitation. Pour que cette commande fonctionne sur un système Linux, vous devez probablement créer le répertoire dans le cadre de la commande (par exemple,`mkdir ~/downloads`).

Le résultat attendu devrait ressembler à ceci :

```
{
  "download_end": "2025-07-18 08:22:39",
  "download_start": "2025-07-18 08:22:33",
  "download_time": "6s",
  "files_downloaded": 1,
  "local_path": "/Users/your-username/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp",
  "size_downloaded": "181.85 MB",
  "status": "Completed",
  "transfer_id": "gpunet_0_pyt_ckpt[version=21.12.0_amp]"
}
```

Renommez le fichier de point de contrôle pour qu'il corresponde au nom attendu dans le code de notre application lors des étapes ultérieures (aucune extraction n'est nécessaire, car il s'agit d'un point de contrôle PyTorch *.pth.tar standard contenant le dictionnaire d'état du modèle) :

```
mv ~/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp/0.65ms.pth.tar gpunet-0.pth
```

Activez le AWS Common Runtime dans la AWS CLI pour optimiser le débit S3 :

```
aws configure set s3.preferred_transfer_client crt
```

Téléchargez les poids du modèle dans votre compartiment S3 :

```
aws s3 cp gpunet-0.pth s3://${S3_BUCKET_NAME}/gpunet-0.pth
```

Le résultat attendu devrait ressembler à ceci :

```
upload: ./gpunet-0.pth to s3://eks-rt-inference-models-us-east-1-1752722786/
gpunet-0.pth
```

Création du service de modèles

Au cours de cette étape, vous allez configurer une application Web FastAPI pour la classification d'images accélérée par GPU à l'aide du modèle de vision -0. GPUNet L'application télécharge les pondérations des modèles depuis Amazon S3 au moment de l'exécution, extrait l'architecture du modèle depuis le référentiel NVIDIA pour la mise en cache et télécharge les étiquettes de ImageNet classe via HTTP. L'application inclut des transformations de prétraitement d'image et expose deux points de terminaison : un point de terminaison GET racine pour la vérification du statut et un point de `/predict` terminaison POST qui accepte une URL d'image.

Nous diffusons le modèle à l'aide de FastAPI PyTorch, en chargeant les poids depuis Amazon S3 au moment de l'exécution dans une configuration conteneurisée pour un prototypage rapide et un déploiement Kubernetes. Pour d'autres méthodes telles que le traitement par lots optimisé ou les moteurs à haut débit, voir Serving ML Models.

Pour créer l'application

Créez un répertoire pour les fichiers de votre application, par exemple`model-testing`, puis remplacez-le par des répertoires et ajoutez le code suivant à un nouveau fichier nommé `app.py` :

```
import os
import torch
import json
import requests
from fastapi import FastAPI, HTTPException
from PIL import Image
from io import BytesIO, StringIO
import torchvision.transforms as transforms
from torch.nn.functional import softmax
import warnings
```

```
from contextlib import redirect_stdout, redirect_stderr
import argparse
import boto3
app = FastAPI()

# Suppress specific warnings from the model code (quantization is optional and unused
 here)
warnings.simplefilter("ignore", UserWarning)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load model code from cache (if present)
# Use backed cache directory
torch.hub.set_dir('/cache/torch/hub')

# Allowlist for secure deserialization (handles potential issues in older checkpoints)
torch.serialization.add_safe_globals([argparse.Namespace])
# Load the model architecture only on container startup (changed to pretrained=False)
# Precision (FP32 for full accuracy, could be 'fp16' for speed on Ampere+ GPUs)
with redirect_stdout(StringIO()), redirect_stderr(StringIO()):
    gpunet = torch.hub.load('NVIDIA/DeepLearningExamples:torchhub', 'nvidia_gpunet',
 pretrained=False, model_type='GPUNet-0', model_math='fp32')

# Download weights from S3 if not present, then load them
model_path = os.getenv('MODEL_PATH', '/cache/torch/hub/checkpoints/gpunet-0.pth')
os.makedirs(os.path.dirname(model_path), exist_ok=True)  # Ensure checkpoints dir
 exists
if not os.path.exists(model_path):
    s3 = boto3.client('s3')
    s3.download_file(os.getenv('S3_BUCKET_NAME'), 'gpunet-0.pth', model_path)
checkpoint = torch.load(model_path, map_location=device, weights_only=True)
gpunet.load_state_dict(checkpoint['state_dict'])
# Move to GPU/CPU
gpunet.to(device)
gpunet.eval()

# Preprocessing
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

```
# Load ImageNet labels
labels_url = "https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json"
response = requests.get(labels_url)
json_data = json.loads(response.text)
labels = [json_data[str(i)][1].replace('_', ' ') for i in range(1000)]

# Required, FastAPI root
@app.get("/")
async def hello():
    return {"status": "hello"}

# Serve model requests
@app.post("/predict")
async def predict(image_url: str):
    try:
        response = requests.get(image_url)
        response.raise_for_status()
        img = Image.open(BytesIO(response.content)).convert("RGB")
        input_tensor = preprocess(img).unsqueeze(0).to(device)

        with torch.no_grad():
            output = gpunet(input_tensor)

        probs = softmax(output, dim=1)[0]
        top5_idx = probs.topk(5).indices.cpu().numpy()
        top5_probs = probs.topk(5).values.cpu().numpy()

        results = [{ "label": labels[idx], "probability": float(prob) } for idx, prob
 in zip(top5_idx, top5_probs)]

        return {"predictions": results}
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

Créez le Dockerfile

Le Dockerfile suivant crée une image de conteneur pour notre application en utilisant le GPUNet modèle du référentiel NVIDIA Deep Learning Examples for Tensor Cores. GitHub

Nous réduisons la taille de l'image du conteneur en utilisant une PyTorch base réservée à l'exécution, en installant uniquement les packages essentiels avec nettoyage du cache, en précachant le code du modèle, et en évitant de « modifier » les poids dans l'image du conteneur afin de permettre des

extractions et des mises à jour plus rapides. Pour en savoir plus, consultez la section Réduction de la taille des images des conteneurs.

Dans le même répertoire que`app.py`, créez `Dockerfile` :

```
FROM pytorch/pytorch:2.4.0-cuda12.4-cudnn9-runtime

# Install required system packages required for git cloning
RUN apt-get update && apt-get install -y git && rm -rf /var/lib/apt/lists/*

# Install application dependencies
RUN pip install --no-cache-dir fastapi uvicorn requests pillow boto3 timm==0.5.4

# Pre-cache the GPUNet code from Torch Hub (without weights)
# Clone the repository containing the GPUNet code
RUN mkdir -p /cache/torch/hub && \
    cd /cache/torch/hub && \
    git clone --branch torchhub --depth 1 https://github.com/NVIDIA/
DeepLearningExamples NVIDIA_DeepLearningExamples_torchhub

COPY app.py /app/app.py

WORKDIR /app

CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "80"]
```

Tester l'application

À partir du même répertoire que votre `app.py` et`Dockerfile`, créez l'image du conteneur pour l'application d'inférence, en ciblant AMD64 l'architecture :

```
docker build --platform linux/amd64 -t gpunet-inference-app .
```

Définissez des variables d'environnement pour vos AWS informations d'identification, et éventuellement un jeton de AWS session. Par exemple :

```
export AWS_REGION="us-east-1"
export AWS_ACCESS_KEY_ID=ABCEXAMPLESCUJFEIELSMUHHAZ
export AWS_SECRET_ACCESS_KEY=123EXAMPLEMZREoQXr8XkiicsOgWDQ5TpUsq0/Z
```

Exécutez le conteneur localement, en injectant des AWS informations d'identification en tant que variables d'environnement pour l'accès à S3. Par exemple :

```
docker run --platform linux/amd64 -p 8080:80 \
  -e S3_BUCKET_NAME=${S3_BUCKET_NAME} \
  -e AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} \
  -e AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY} \
  -e AWS_DEFAULT_REGION=${AWS_REGION} \
  gpunet-inference-app
```

Le résultat attendu devrait ressembler à ceci :

```
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
```

Dans une nouvelle fenêtre de terminal, testez le point de terminaison d'inférence en envoyant un exemple de requête POST avec une URL d'image publique comme paramètre de requête :

```
curl -X POST "http://localhost:8080/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

Le résultat attendu doit être une réponse JSON avec les 5 meilleures prédictions, similaire à celle-ci (les étiquettes et les probabilités réelles peuvent varier légèrement en fonction de la précision de l'image et du modèle) :

```
{"predictions":[{"label":"desk","probability":0.28885871171951294},
{"label":"laptop","probability":0.24679335951805115},
{"label":"notebook","probability":0.08539070934057236},
{"label":"library","probability":0.030645888298749924},
{"label":"monitor","probability":0.02989606373012066}]}
```

Quittez l'application en appuyant sur « Ctrl + C ».

Transférez le conteneur vers Amazon ECR

Au cours de cette étape, nous téléchargeons l'image du conteneur pour le service modèle GPUNet -0 sur Amazon Elastic Container Registry (ECR), afin de la rendre disponible pour le déploiement sur Amazon EKS. Ce processus implique la création d'un nouveau référentiel ECR pour stocker l'image, l'authentification avec ECR, puis le balisage et le transfert de l'image du conteneur vers notre registre.

Tout d'abord, revenez au répertoire dans lequel vous avez défini vos variables d'environnement au début de ce guide. Par exemple :

```
cd ..
```

Créez un référentiel dans Amazon ECR :

```
aws ecr create-repository --repository-name gpunet-inference-app --region ${AWS_REGION}
```

Connectez-vous à Amazon ECR :

```
aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS --
password-stdin ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
```

Le résultat attendu devrait ressembler à ceci :

```
Login Succeeded
```

Marquez l'image :

```
docker tag gpunet-inference-app:latest ${AWS_ACCOUNT_ID}.dkr.ecr.
${AWS_REGION}.amazonaws.com/gpunet-inference-app:latest
```

Transférez l'image vers votre référentiel Amazon ECR :

```
docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
```

Cette dernière étape prend plusieurs minutes.

7. (Facultatif) Exposez le modèle de service

Au cours de cette étape, vous allez exposer votre service de modèle d'inférence en temps réel en externe sur Amazon EKS à l'aide du AWS Load Balancer Controller (LBC). Cela implique de configurer le LBC, de monter les pondérations des modèles depuis Amazon S3 sous forme de volume persistant à l'aide du pilote CSI Mountpoint S3, de déployer un module d'application accéléré par GPU, de créer un service et une entrée pour configurer un Application Load Balancer (ALB) et de tester le point de terminaison.

Vérifiez d'abord l'association Pod Identity pour le AWS LBC, en vous assurant que le compte de service est correctement lié au rôle IAM requis :

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --namespace kube-system
 --service-account-name aws-load-balancer-controller
```

Le résultat attendu devrait ressembler à ceci :

```
ASSOCIATION ARN                                                    NAMESPACE    SERVICE
 ACCOUNT NAME        IAM ROLE ARN    OWNER ARN
arn:aws:eks:us-east-1:143095308808:podidentityassociation/eks-rt-inference-us-
east-1/a-buavluu2wp1jropya    kube-system     aws-load-balancer-controller
 arn:aws:iam::143095308808:role/AmazonEKSLoadBalancerControllerRole
```

Marquez le groupe de sécurité de votre cluster

Le AWS Load Balancer Controller ne prend en charge qu'un seul groupe de sécurité avec la clé de balise `karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"` pour la sélection du groupe de sécurité par Karpenter. Lors de la création d'un cluster avec eksctl, le groupe de sécurité du cluster par défaut (qui possède la `"kubernetes.io/cluster/<cluster-name>: owned"` balise) n'est pas automatiquement étiqueté avec des `karpenter.sh/discovery` balises. Cette balise est essentielle pour que Karpenter découvre et attache ce groupe de sécurité aux nœuds qu'il approvisionne. L'attachement de ce groupe de sécurité garantit la compatibilité avec le AWS Load Balancer Controller (LBC), ce qui lui permet de gérer automatiquement les règles de trafic entrant pour les services exposés via Ingress, tels que le service modèle décrit dans ces étapes.

Exportez l'ID VPC de votre cluster :

```
CLUSTER_VPC_ID="$(aws eks describe-cluster --name ${EKS_CLUSTER_NAME} --query
 cluster.resourcesVpcConfig.vpcId --output text)"
```

Exportez le groupe de sécurité par défaut pour votre cluster :

```
CLUSTER_SG_ID="$(aws ec2 describe-security-groups --filters Name=vpc-id,Values=
$CLUSTER_VPC_ID Name=tag-key,Values=kubernetes.io/cluster/${EKS_CLUSTER_NAME} --query
 'SecurityGroups[].[GroupId]' --output text)"
```

Ajoutez la `karpenter.sh/discovery` balise au groupe de sécurité du cluster par défaut. Cela permettra à nos EC2 NodeClass sélecteurs de CPU et de GPU de l'utiliser :

```
aws ec2 create-tags --resources ${CLUSTER_SG_ID} --tags Key=karpenter.sh/
discovery,Value=${EKS_CLUSTER_NAME}
```

Vérifiez que le tag a été ajouté :

```
aws ec2 describe-security-groups --group-ids ${CLUSTER_SG_ID} --query
 "SecurityGroups[].Tags"
```

Parmi les résultats, vous devriez voir ce qui suit avec le tag et le nom de votre cluster. Par exemple :

```
{
  "Key": "karpenter.sh/discovery",
  "Value": "eks-rt-inference-us-east-1"
}
```

Configuration du AWS Load Balancer Controller (LBC)

Le AWS LBC est essentiel pour gérer le trafic entrant vers les charges de AI/ML travail sur Amazon EKS, en garantissant l'accès aux points de terminaison d'inférence ou aux pipelines de traitement des données. En s'intégrant aux équilibreurs de charge d' AWS application (ALB) et aux équilibreurs de charge réseau (NLB), le LBC achemine dynamiquement le trafic vers des applications conteneurisées, telles que celles qui exécutent de grands modèles linguistiques, des modèles de vision par ordinateur ou des services d'inférence en temps réel. Comme nous avons déjà créé le compte de service et l'association Pod Identity lors de la création du cluster, nous les avons définis pour qu'ils correspondent `serviceAccount.name` à ce qui est défini dans notre configuration de cluster (`aws-load-balancer-controller`).

Ajoutez le référentiel de cartes eks-charts Helm AWS appartenant à eks-charts :

```
helm repo add eks https://aws.github.io/eks-charts
```

Actualisez vos référentiels Helm locaux avec les graphiques les plus récents :

```
helm repo update eks
```

Déployez le AWS LBC à l'aide de Helm, en spécifiant le nom du cluster EKS et en faisant référence au compte de service pré-créé :

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=${EKS_CLUSTER_NAME} \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

Le résultat attendu devrait ressembler à ceci :

```
NAME: aws-load-balancer-controller
LAST DEPLOYED: Wed Jul 9 15:03:31 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

Monter le modèle dans un volume persistant

Au cours de cette étape, vous allez monter les pondérations des modèles à partir de votre compartiment Amazon S3 à l'aide d'un PersistentVolume (PV) soutenu par le pilote CSI Mountpoint for Amazon S3. Cela permet aux pods Kubernetes d'accéder aux objets S3 sous forme de fichiers locaux, éliminant ainsi les téléchargements gourmands en ressources vers des espaces de stockage éphémères ou des conteneurs d'initialisation, ce qui est idéal pour les modèles volumineux de plusieurs gigaoctets.

Le PV monte l'intégralité de la racine du compartiment (aucun chemin n'est spécifié dans`volumeAttributes`), prend en charge l'accès simultané en lecture seule par plusieurs modules et expose des fichiers tels que les poids du modèle (`/models/gpunet-0.pth`) à l'intérieur du conteneur à des fins d'inférence. Cela garantit que le « téléchargement » de secours dans notre application (`app.py`) ne se déclenche pas car le fichier existe via le montage. En découplant le modèle de l'image du conteneur, cela permet un accès partagé et des mises à jour indépendantes des versions du modèle sans avoir à reconstruire l'image.

Créez le PersistentVolume (PV)

Créez une ressource PersistentVolume (PV) pour monter le compartiment S3 contenant les poids de votre modèle, en permettant un accès en lecture seule à plusieurs pods sans télécharger de fichiers lors de l'exécution :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: s3-model-pv
spec:
  capacity:
```

```
    storage: 5Gi  # Ignored by the driver; can be any value
  accessModes:
    - ReadOnlyMany  # Read only
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""  # Required for static provisioning
  claimRef:
    namespace: default  # Adjust if you prefer a different namespace
    name: s3-model-pvc
  mountOptions:
    - allow-other  # Enables multi-user access (useful for non-root pods)
    - region ${AWS_REGION} # Optional, include if your bucket is in a different region
 than the cluster
  csi:
    driver: s3.csi.aws.com
    volumeHandle: gpunet-model-volume  # Must be unique across all PVs
    volumeAttributes:
      bucketName: ${S3_BUCKET_NAME}
EOF
```

Créez le PersistentVolumeClaim (PVC)

Créez un PersistentVolumeClaim (PVC) à lier au PV, en demandant un accès en lecture seule aux données du modèle S3 monté :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: s3-model-pvc
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ""  # Required for static provisioning
  resources:
    requests:
      storage: 5Gi  # Ignored, match PV capacity
  volumeName: s3-model-pv  # Bind to the PV created above
EOF
```

Déployer l'application

Déployez l'application d'inférence sous la forme d'un déploiement Kubernetes, en montant le volume persistant soutenu par S3 pour l'accès au modèle, en appliquant des sélecteurs de nœuds GPU

et des tolérances, et en définissant des variables d'environnement pour le chemin du modèle. Ce déploiement définit le chemin du modèle (env var of"`/models/gpunet-0.pth`"), de sorte que notre application (in`app.py`) utilisera ce chemin par défaut. Lorsque le volume du déploiement est monté sur `/models` (lecture seule), le téléchargement du modèle ne se déclenchera pas si le fichier est déjà présent via le PVC.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpunet-inference-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpunet-inference-app
  template:
    metadata:
      labels:
        app: gpunet-inference-app
    spec:
      tolerations:
      - key: "nvidia.com/gpu"
        operator: "Exists"
        effect: "NoSchedule"
      nodeSelector:
        role: gpu-worker
      containers:
      - name: inference
        image: ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
        ports:
        - containerPort: 80
        env:
        - name: MODEL_PATH
          value: "/models/gpunet-0.pth"
        resources:
          limits:
            nvidia.com/gpu: 1
          requests:
            nvidia.com/gpu: 1
        volumeMounts:
        - name: model-volume
```

```
          mountPath: /models
          readOnly: true
      volumes:
      - name: model-volume
        persistentVolumeClaim:
          claimName: s3-model-pvc
EOF
```

Karpenter mettra quelques minutes à provisionner un nœud GPU s'il n'en existe pas déjà un. Vérifiez que le module d'inférence est dans un état « en cours d'exécution » :

```
kubectl get pods -l app=gpunet-inference-app
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                                READY   STATUS    RESTARTS   AGE
gpunet-inference-app-5d4b6c7f8-abcde        1/1     Running   0          2m
```

Exposez le service avec Ingress et Load Balancer

Créez un service ClusterIP pour exposer le déploiement d'inférence en interne au sein du cluster EKS, en ciblant le port de l'application :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: gpunet-model-service
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: gpunet-inference-app
EOF
```

Créez une ressource Ingress pour approvisionner un Application Load Balancer (ALB) connecté à Internet via le LBC, en acheminant le trafic externe vers AWS le service d'inférence :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
metadata:
  name: gpunet-model-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: gpunet-model-service
            port:
              number: 80
EOF
```

Attendez quelques minutes pour que l'Application Load Balancer (ALB) termine le provisionnement. Surveillez l'état des ressources d'entrée pour confirmer que l'ALB a été provisionné :

```
kubectl get ingress gpunet-model-ingress
```

La sortie attendue devrait ressembler à ceci (avec le champ ADDRESS rempli) :

```
NAME                   CLASS   HOSTS   ADDRESS
 PORTS   AGE
gpunet-model-ingress   alb     *       k8s-default-gpunetmo-183de3f819-516310036.us-
east-1.elb.amazonaws.com   80      6m58s
```

Extrayez et exportez le nom d'hôte ALB à partir de l'état d'entrée pour l'utiliser lors de tests ultérieurs :

```
export ALB_HOSTNAME=$(kubectl get ingress gpunet-model-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Testez le Model Service

Validez le point de terminaison d'inférence exposé en envoyant une requête POST avec un exemple d'URL d'image (par exemple, à partir du jeu de données COCO), en simulant une prédiction en temps réel :

```
curl -X POST "http://${ALB_HOSTNAME}/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

Le résultat attendu doit être une réponse JSON avec les 5 meilleures prédictions, similaire à celle-ci (les étiquettes et les probabilités réelles peuvent varier légèrement en fonction de la précision de l'image et du modèle) :

```
{"predictions":[{"label":"desk","probability":0.2888975441455841},
{"label":"laptop","probability":0.2464350312948227},
{"label":"notebook","probability":0.08554483205080032},
{"label":"library","probability":0.030612602829933167},
{"label":"monitor","probability":0.029896672815084457}]}
```

Vous pouvez éventuellement continuer à tester d'autres images dans une nouvelle requête POST. Par exemple :

```
http://images.cocodataset.org/test-stuff2017/000000024309.jpg
http://images.cocodataset.org/test-stuff2017/000000028117.jpg
http://images.cocodataset.org/test-stuff2017/000000006149.jpg
http://images.cocodataset.org/test-stuff2017/000000004954.jpg
```

Conclusion

Dans ce guide, vous allez configurer un cluster Amazon EKS optimisé pour les charges de travail d'inférence en temps réel accélérées par GPU. Vous avez provisionné un cluster avec des EC2 instances G5, installé le pilote CSI Mountpoint S3, l'agent EKS Pod Identity, l'agent de surveillance des nœuds EKS, l'AMI Bottlerocket, le AWS Load Balancer Controller (LBC) et Karpenter pour gérer le processeur et le GPU. NodePools Vous avez utilisé le plug-in NVIDIA Device pour activer la planification du GPU et vous avez configuré S3 avec PersistentVolume et PersistentVolumeClaim pour l'accès aux modèles. Vous avez validé la configuration en déployant un exemple de pod GPU, en configurant l'accès au modèle NVIDIA GPUNet-0 sur Amazon S3, en activant l'initialisation du pod et en exposant le service d'inférence via Application Load Balancer. Pour utiliser pleinement votre cluster, configurez l'agent de surveillance des nœuds EKS avec réparation automatique. Veillez à effectuer des tests de référence, notamment des évaluations des performances, de la latence et du débit du GPU afin d'optimiser les temps de réponse. Pour en savoir plus, consultez la section Utilisation des outils de surveillance et d'observabilité pour vos charges AI/ML de travail.

Nettoyage

Pour éviter d'encourir de futurs frais, vous devez supprimer manuellement la CloudFormation pile associée pour supprimer toutes les ressources créées au cours de ce guide, y compris le réseau VPC.

Supprimez la CloudFormation pile en utilisant le `--wait` drapeau avec eksctl :

```
eksctl delete cluster --region ${AWS_REGION} --name ${EKS_CLUSTER_NAME} --wait
```

À la fin, vous devriez voir le résultat de réponse suivant :

```
2025-07-29 13:03:55 [#]  all cluster resources were deleted
```

Supprimez le compartiment Amazon S3 créé au cours de ce guide à l'aide de la console Amazon S3.

Configuration du cluster Amazon EKS pour les charges AI/ML de travail

Cette section est conçue pour vous aider à configurer des clusters Amazon EKS optimisés pour les AI/ML charges de travail. Vous y trouverez des conseils sur l'exécution de conteneurs accélérés par GPU optimisés pour Linux et Windows AMIs, sur la configuration de clusters d'entraînement avec Elastic Fabric Adapter (EFA) pour un réseau à hautes performances et sur la création de clusters d'inférence avec des instances AWS Inferentia, y compris les conditions préalables, les procédures et les considérations relatives au déploiement. step-by-step

Rubriques

- Exécuter des conteneurs accélérés par GPU (Linux activé) EC2
- Exécutez des conteneurs accélérés par GPU (Windows sur G-Series) EC2
- Organisez une formation de machine learning sur Amazon EKS avec Elastic Fabric Adapter
- Utiliser des instances AWS Inferentia avec Amazon EKS pour le Machine Learning

Exécuter des conteneurs accélérés par GPU (Linux activé) EC2

Les systèmes Amazon Linux accélérés optimisés pour Amazon EKS AMIs sont basés sur le standard Amazon Linux optimisé pour Amazon EKS AMIs. Pour plus de détails à ce AMIs sujet, voirthe section

called "Amazon EKS a optimisé Amazon Linux accéléré AMIs". Le texte suivant décrit comment activer les charges de travail AWS basées sur les neurones.

Pour activer les charges de travail basées AWS sur Neuron (accélérateur ML)

Pour en savoir plus sur les charges de travail d'entraînement et d'inférence à l'aide de Neuron dans Amazon EKS, consultez les références suivantes :

- Conteneurs - Kubernetes - Commencer à utiliser la documentation Neuron AWS
- Formation à AWS Neuron EKS Samples sur GitHub
- Déployez des charges de travail d'inférence ML AWSInferentia sur Amazon EKS

La procédure suivante décrit comment exécuter une charge de travail sur une instance basée sur un GPU avec l'accélérateur optimisé Amazon EKS AMIs.

1. Une fois que vos nœuds GPU ont rejoint votre cluster, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes DaemonSet sur votre cluster. *vX.X.X*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

2. Vous pouvez vérifier que vos nœuds sont allouables à l' GPUs aide de la commande suivante.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

3. Créez un fichier nommé `nvidia-smi.yaml` avec les contenus suivants. Remplacez-le par le *tag* tag de votre choix pour nvidia/cuda. Ce manifeste lance un conteneur NVIDIA CUDA qui s'exécute `nvidia-smi` sur un nœud.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:tag
```

```
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

4. Appliquez le manifeste ci-dessus avec la commande suivante.

```
kubectl apply -f nvidia-smi.yaml
```

5. Une fois que le Pod a fini de fonctionner, consultez ses journaux à l'aide de la commande suivante.

```
kubectl logs nvidia-smi
```

L'exemple qui suit illustre un résultat.

```
Mon Aug  6 20:23:31 20XX
+-----------------------------------------------------------------------------+
| NVIDIA-SMI XXX.XX                 Driver Version: XXX.XX                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0    47W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Exécutez des conteneurs accélérés par GPU (Windows sur G-Series) EC2

> **Important**
>
> Le plugin Kubernetes Device pour DirectX by TensorWorks est un outil tiers qui n'est pas approuvé, pris en charge ou maintenu par. AWS AWS n'assume aucune responsabilité quant à la sécurité, à la fiabilité ou aux performances de ce plugin.

Découvrez comment exécuter des charges de travail de conteneurs Windows accélérées par GPU sur Amazon EKS (Elastic Kubernetes Service) à l'aide de GPUs NVIDIA avec le plug-in Kubernetes Device pour DirectX by. TensorWorks Pour plus d'informations, voir Plug-in d'appareil Kubernetes pour DirectX.

Il existe deux approches principales pour configurer l'accélération GPU pour vos conteneurs Windows :

- Option 1 : créez une AMI EKS personnalisée optimisée pour Windows avec les pilotes de GPU requis préinstallés.
  - Utilisez cette approche lorsque vous avez besoin d'un environnement préconfiguré cohérent prêt à exécuter des conteneurs Windows accélérés par GPU, et que vous êtes en mesure d'investir des efforts supplémentaires pour créer et gérer l'AMI personnalisée.
- Option 2 : installez les pilotes GPU nécessaires sur vos nœuds de travail EKS après le lancement de votre instance.
  - Utilisez cette approche lorsque vous souhaitez simplifier le processus de configuration et que cela ne vous dérange pas d'installer les pilotes GPU sur chaque nouveau nœud de travail. Plus adapté à un environnement de développement lorsque vous évaluez ou prototypez des charges de travail accélérées par GPU.

Les deux approches peuvent être mises à profit en suivant les étapes décrites dans ce guide.

Considérations

Ce guide explique comment installer et configurer l'accélération GPU pour vos conteneurs Windows à l'aide de NVIDIA GPUs, des pilotes NVIDIA GRID et du plug-in Kubernetes Device pour DirectX by. TensorWorks Les étapes ont été testées et vérifiées afin de fournir une accélération GPU pour les charges de travail de vos conteneurs Windows sur Amazon EKS. Consultez the section called

"Limitations connues" pour plus d'informations sur les pilotes et les plug-ins d'appareils compatibles. Avant de poursuivre, notez ce qui suit :

- Seuls les types d'instances de la famille G dotés de pilotes NVIDIA GRID ont été testés et vérifiés pour fonctionner avec ce guide. Bien que d'autres types d'instances et combinaisons de pilotes soient également capables d'exécuter des conteneurs Windows accélérés par GPU, ils peuvent nécessiter des étapes de configuration supplémentaires non décrites dans ce guide.
- Seules les charges de travail basées sur DirectX ont été testées et vérifiées pour fonctionner avec ce guide. Bien que d'autres GPU APIs tels qu'OpenGL, Vulkan et OpenCL soient potentiellement compatibles pour exécuter des conteneurs Windows accélérés par GPU, ils peuvent nécessiter des étapes de configuration supplémentaires non décrites dans ce guide.
- Il existe certaines limites connues à prendre en compte avant d'exécuter des conteneurs Windows accélérés par GPU. Veuillez consulter la the section called "Limitations connues" section pour plus d'informations.

Prérequis

Pour activer l'accélération GPU pour vos conteneurs Windows sur Amazon EKS, vous devez préparer les exigences suivantes avant de continuer :

- Lancez un cluster Amazon EKS avec Kubernetes v1.27 ou version ultérieure.
- Provisionnez les nœuds Windows avec Windows Server 2022 ou une version ultérieure.
- Provisionnez des nœuds Windows appartenant à la famille de types d'instances G, tels que G4 ou G5.
- Provisionnez des nœuds Windows avec un environnement d'exécution de conteneur avec containerd ou. `1.7.x` `2.x.x` (Consultez the section called "Obtenir des informations sur la version" pour vérifier la version containerd dans votre AMI optimisée Amazon EKS.)

Installez le pilote GPU sur chaque nœud Windows Windows

Pour installer les pilotes NVIDIA GRID sur vos nœuds de travail EKS, suivez les étapes décrites dans la section Pilotes NVIDIA pour votre EC2 instance Amazon. Accédez à Options d'installation - Option 3 : pilotes GRID et suivez les étapes d'installation.

Installation pour Windows Server Core

Pour Windows Server Core, qui n'est pas compatible avec les ordinateurs de bureau, installez les pilotes NVIDIA GRID de manière silencieuse à l'aide des commandes suivantes :

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

Vérifiez votre installation

Exécutez la PowerShell commande suivante pour afficher les informations de diagnostic relatives GPUs à l'instance :

```
nvidia-smi
```

Cette commande affiche la version du pilote NVIDIA, ainsi que des informations sur le matériel du GPU. Assurez-vous que le résultat de cette commande correspond à la version du pilote NVIDIA GRID que vous vous attendiez à installer.

Déployez le plug-in du périphérique GPU sur chaque nœud

Pour permettre la découverte et l'exposition des ressources du GPU aux conteneurs de vos nœuds Windows, vous aurez besoin d'un plug-in pour appareil. Déployez le plug-in DirectX Device de Tensorworks sur chaque nœud de travail en l'exécutant comme DaemonSet dans votre cluster EKS. Suivez le guide d'installation spécifié dans le fichier README.md, qui comportera les étapes suivantes. Il est recommandé de :

- Déployez le plug-in de l'appareil dans l'espace de `kube-system` noms.
- Définissez des limites de ressources appropriées pour DaemonSet vous assurer qu'il ne consomme pas trop de ressources sur vos nœuds.

> **Note**
>
> Le plug-in de périphérique s' DaemonSet exécutera sur chaque nœud en tant que conteneur de processus hôte avec des privilèges élevés. Il est recommandé d'implémenter des contrôles RBAC pour en restreindre l'accès DaemonSet afin que seuls les utilisateurs autorisés puissent exécuter des commandes privilégiées.

Lors de l'exécution de conteneurs accélérés par GPU, le plug-in de l'appareil prend en charge deux modes :

- Mode mono-locataire : ce mode dédie toutes les ressources du GPU à un seul conteneur sur l'instance. Installez les plug-ins de l'appareil avec support mono-locataire à l'aide de la commande suivante. Consultez README.md pour plus d'informations.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
  ```

- Mode multi-tenancy : ce mode permet de partager les ressources GPU entre plusieurs conteneurs de l'instance. Installez les plug-ins de l'appareil compatibles avec le multi-tenant à l'aide de la commande suivante. Consultez README.md pour plus d'informations.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
  ```

  Vous pouvez également utiliser a ConfigMap pour spécifier la mutualisation.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
  ```

Vérification du déploiement du plug-in de l'appareil

Après avoir déployé le plug-in de périphérique, remplacez `<namespace>` et exécutez la commande suivante pour vérifier que le plug-in de périphérique DirectX fonctionne correctement sur tous vos nœuds Windows.

```
kubectl get ds device-plugin-wddm -n <namespace>
```

Vérifier que les conteneurs sont prêts à être déployés

Une fois que le plug-in de l'appareil DaemonSet est en cours d'exécution sur les nœuds de travail Windows alimentés par le GPU, utilisez la commande suivante pour vérifier que chaque nœud est allouable. GPUs Le nombre correspondant doit correspondre au nombre de périphériques DirectX sur chaque nœud.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/
display"
```

Exécution de conteneurs Windows avec accélération par GPU

Avant de lancer vos pods, spécifiez le nom de la ressource `directx.microsoft.com/display` dans `.spec.containers[].resources`. Cela indiquera que vos conteneurs nécessitent des fonctionnalités compatibles avec le GPU et `kube-scheduler` tenteront de placer vos pods sur votre nœud Windows préconfiguré avec les ressources GPU disponibles.

À titre d'exemple, consultez l'exemple de commande ci-dessous qui lance une `Job` simulation de Monte Carlo pour estimer la valeur de pi. Cet exemple provient du GitHub référentiel Kubernetes Device Plugins for DirectX, qui propose plusieurs exemples que vous pouvez exécuter pour tester les capacités du GPU de votre nœud Windows.

```
cat <<EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: example-cuda-montecarlo-wddm
spec:
  template:
    spec:
      containers:
      - name: example-cuda-montecarlo-wddm
        image: "index.docker.io/tensorworks/example-cuda-montecarlo:0.0.1"
        resources:
          limits:
            directx.microsoft.com/display: 1
      nodeSelector:
        "kubernetes.io/os": windows
      restartPolicy: Never
  backoffLimit: 0
EOF
```

Limitations connues

Tous GPUs sont utilisables

Tout ce qui se trouve GPUs sur l'instance sera utilisable par chaque conteneur en cours d'exécution sur l'hôte, même lorsque vous demandez un nombre spécifique de GPUs pour un conteneur donné. En outre, le comportement par défaut est que tous les conteneurs exécutés sur l'hôte utiliseront le GPU avec un indice 0, même s'il y en a plusieurs GPUs disponibles sur le nœud. Ainsi, pour que les tâches multi-GPU fonctionnent correctement, vous devez désigner explicitement le périphérique GPU spécifique à utiliser dans le code de votre application.

L'implémentation exacte pour allouer un périphérique à utiliser pour l'application dépend du langage de programmation ou du framework que vous utilisez. Par exemple, si vous utilisez la programmation CUDA, pour sélectionner un GPU spécifique, vous pouvez spécifier explicitement le périphérique à utiliser dans le code de votre application à l'aide de la fonction cudaSetDevice().

La nécessité de spécifier explicitement le périphérique est due à un problème connu affectant les conteneurs Windows. Vous pouvez suivre les progrès réalisés dans la résolution de ce problème dans le numéro #333 de Microsoft/Windows-Containers. Le tableau suivant représente une représentation visuelle et un exemple pratique de ce comportement d'allocation de GPU.

Imaginons un scénario dans lequel il existe un seul nœud Windows de type EC2 instance`g4dn.12xlarge`, qui en contient quatre GPUs. Imaginons un scénario dans lequel trois pods sont lancés sur cette instance. Le tableau montre que quel que soit le nombre de requêtes GPUs demandées par chaque conteneur, les trois pods ont accès aux quatre GPUs de l'instance et utilisent par défaut le GPU avec un index de périphérique 0.

Pod	Demandé GPUs	Accès réel au GPU	Utilisation du GPU par défaut	Indices GPU disponibles	Nombre total d'instances GPUs
Capsule 1	1 CARTE GRAPHIQUE	Tous les 4 GPUs	GPU avec indice 0	0, 1, 2, 3	4
Capsule 2	2 GPUs	Tous les 4 GPUs	GPU avec indice 0	0, 1, 2, 3	4
Capsule 3	1 CARTE GRAPHIQUE	Tous les 4 GPUs	GPU avec indice 0	0, 1, 2, 3	4

Support des plug-ins pour appareils Kubernetes

L'implémentation officielle par NVIDIA du plugin pour appareil Kubernetes n'est pas compatible avec Windows. Vous pouvez suivre la progression de l'ajout du support officiel pour Windows dans le numéro #419 de Nvidia/K8 s-device-plugin .

Limitations des instances de calcul GPU

Selon la configuration de votre AWS compte, vous pouvez avoir des limites de service quant au nombre et au type d'instances de calcul EC2 GPU Amazon que vous pouvez lancer. Si vous avez besoin de capacités supplémentaires, vous pouvez demander une augmentation de quota.

Doit créer une AMI optimisée pour le GPU pour Windows

Aucune AMI optimisée pour le GPU EKS pour Windows ou aucun composant géré EC2 Image Builder n'est fourni par Amazon EKS. Vous devrez suivre les étapes décrites dans ce guide pour créer une AMI EKS optimisée pour Windows personnalisée avec les pilotes GPU requis préinstallés, ou installer les pilotes GPU nécessaires sur vos nœuds de travail EKS après le lancement de vos instances.

Inferentia et Trainium ne sont pas pris en charge

AWS Les charges de travail basées sur Inferentia et AWS Trainium ne sont pas prises en charge sous Windows.

Organisez une formation de machine learning sur Amazon EKS avec Elastic Fabric Adapter

Cette rubrique décrit comment intégrer Elastic Fabric Adapter (EFA) aux pods déployés dans votre cluster Amazon EKS. Elastic Fabric Adapter (EFA) est une interface réseau pour les instances EC2 Amazon qui vous permet d'exécuter des applications nécessitant des niveaux élevés de communications entre nœuds à grande échelle. AWS Son interface matérielle sur mesure de contournement du système d'exploitation améliore les performances des communications entre instances, ce qui est essentiel à la mise à l'échelle de ces applications. Avec EFA, les applications de calcul à haute performance (HPC) utilisant l'interface MPI (Message Passing Interface) et les applications Machine Learning (ML) utilisant la bibliothèque de communications collectives (NCCL) de NVIDIA peuvent atteindre des milliers de ou. CPUs GPUs Vous bénéficiez ainsi des performances applicatives des clusters HPC sur site avec l'élasticité et la flexibilité à la demande du AWS cloud. L'intégration d'EFA aux applications exécutées sur des clusters Amazon EKS permet de réduire

le temps nécessaire à l'exécution des applications d'entraînement distribué à grande échelle sans avoir à ajouter d'autres instances à votre cluster. Pour plus d'informations sur l'EFA, consultez Elastic Fabric Adapter.

Types d'instances avec EFA

Le plugin AWS EFA Kubernetes Device prend en charge tous les types d' EC2 instances Amazon dotés d'EFA. Pour consulter la liste de tous les types d'instances dotés de l'EFA, consultez la section Types d'instances pris en charge dans le guide de EC2 l'utilisateur Amazon. Toutefois, pour exécuter rapidement des applications ML, nous recommandons qu'une instance dispose de puces d'accélération matérielle telles que les puces NVidia GPUs, AWS Inferentia ou AWS Trainium, en plus de l'EFA. Pour consulter la liste des types d'instances dotés de puces d'accélération matérielle et d'EFA, consultez la section Accelerated computing dans le guide de EC2 l'utilisateur Amazon.

Lorsque vous comparez les types d'instance pour choisir entre eux, prenez en compte le nombre de cartes réseau EFA disponibles pour ce type d'instance ainsi que le nombre de cartes accélératrices, la quantité de processeur et la quantité de mémoire. Vous pouvez attribuer un EFA maximum par carte réseau. Un EFA est considéré comme une interface réseau. Pour savoir combien d'EFA sont disponibles pour chaque type d'instance doté d'EFA, consultez la liste des cartes réseau dans le guide de EC2 l'utilisateur Amazon.

Interfaces EFA et EFA uniquement

Un Elastic Fabric Adapter (EFA) est une interface réseau qui combine les fonctionnalités d'un adaptateur réseau élastique (ENA) et d'une interface de contournement du système d'exploitation, alimentée par AWS le protocole Scalable Reliable Datagram (SRD). Les fonctionnalités EFA permettent aux applications de communiquer directement avec le matériel pour un transport à faible latence. Vous pouvez choisir d'accéder uniquement aux fonctionnalités EFA à l'aide d'interfaces EFA uniquement, en limitant les communications aux interfaces situées au sein de la même zone de disponibilité.

Pour créer des nœuds pouvant avoir des interfaces uniquement EFA, vous devez utiliser un modèle de EC2 lancement personnalisé et définir le paramètre sur. `InterfaceType` `efa-only` Dans votre modèle de lancement personnalisé, vous ne pouvez pas configurer la carte `0` réseau sur une interface uniquement EFA, car il s'agit de la carte réseau principale et de l'interface réseau de l'instance. EC2 Vous devez disposer de la version VPC CNI `1.18.5` ou d'une version ultérieure pour les interfaces EFA uniquement. Si vous utilisez Amazon Linux 2, la version ami doit être `v20240928` ou ultérieure pour les interfaces EFA uniquement.

La procédure suivante vous guide pour créer un cluster EKS avec des nœuds `eksctl` dotés d'interfaces NVidia GPUs et EFA. Vous ne pouvez pas l'utiliser `eksctl` pour créer des nœuds et des groupes de nœuds utilisant uniquement des interfaces EFA.

Prérequis

- Un cluster Amazon EKS existant. Si vous n'avez pas de cluster existant, créez-en un à l'aide deMise en route.. Votre cluster doit être déployé dans un VPC disposant d'au moins un sous-réseau privé ayant suffisamment d'adresses IP disponibles dans lequel déployer des nœuds. Le sous-réseau privé doit disposer d'un accès Internet sortant fourni par un appareil externe, tel qu'une passerelle NAT.

  Si vous prévoyez d'utiliser `eksctl` pour créer votre groupe de nœuds, `eksctl` peut également créer un cluster pour vous.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut être identique ou supérieure à une version mineure antérieure ou ultérieure à la version Kubernetes de votre cluster. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called “Configurer kubectl et eksctl”.
- Vous devez avoir installé le plug-in Amazon VPC CNI pour Kubernetes `1.7.10` ou version ultérieure avant de lancer des nœuds de travail prenant en charge plusieurs adaptateurs Elastic Fabric, tels que le ou. `p4d` `p5` Pour plus d'informations sur la mise à jour de votre plugin Amazon VPC CNI pour la version Kubernetes, consultez. the section called “CNI Amazon VPC”
- Pour les instances p6-b200, vous devez utiliser EFA Device Plugin version v0.5.6 ou ultérieure.

> **Important**
>
> Une considération importante requise pour adopter EFA avec Kubernetes est la configuration et la gestion de Huge Pages en tant que ressource dans le cluster. Pour plus d'informations, consultez Gérer Huge Pages dans la documentation Kubernetes. Les EC2 instances Amazon sur lesquelles le pilote EFA est installé préallouent 5 128 pages volumineuses de 2 Mo, que vous pouvez demander comme ressources à utiliser dans le cadre de vos spécifications de travail.

Créer un groupe de nœuds

La procédure suivante vous aide à créer un groupe de nœuds avec un groupe de nœuds `p4d.24xlarge` sauvegardé avec des interfaces EFA et GPUDirect RDMA, et à exécuter un exemple de test NVIDIA Collective Communications Library (NCCL) pour les performances NCCL multi-nœuds à l'aide de. EFAs L'exemple peut être utilisé comme modèle de formation en profondeur distribuée sur Amazon EKS à l'aide de EFAs.

1. Déterminez quels types d' EC2 instances Amazon prenant en charge l'EFA sont disponibles dans la AWS région dans laquelle vous souhaitez déployer des nœuds. *`region-code`*Remplacez-le par la AWS région dans laquelle vous souhaitez déployer votre groupe de nœuds.

   ```
   aws ec2 describe-instance-types --region region-code \
       --filters Name=network-info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   Lorsque vous déployez des nœuds, le type d'instance que vous souhaitez déployer doit être disponible dans la AWS région dans laquelle se trouve votre cluster.

2. Déterminez dans quelles zones de disponibilité le type d'instance que vous souhaitez déployer est disponible. Dans ce didacticiel, le type d'`p5.48xlarge`instance est utilisé et doit être renvoyé dans la sortie pour la AWS région que vous avez spécifiée à l'étape précédente. Lorsque vous déployez des nœuds dans un cluster de production, remplacez-les *`p5.48xlarge`* par n'importe quel type d'instance renvoyé à l'étape précédente.

   ```
   aws ec2 describe-instance-type-offerings --region region-code \
       --location-type availability-zone --filters Name=instance-
   type,Values=p4d.24xlarge,p5.48xlarge \
       --query 'InstanceTypeOfferings[*].Location' --output text
   ```

L'exemple qui suit illustre un résultat.

```
us-west-2a    us-west-2c    us-west-2b
```

Notez les zones de disponibilité renvoyées pour une utilisation ultérieure. Lorsque vous déployez des nœuds sur un cluster, votre VPC doit disposer de sous-réseaux avec des adresses IP disponibles dans l'une des zones de disponibilité renvoyées dans la sortie.

3. Créez un groupe de nœuds à l'aide de`eksctl`. Vous avez besoin d'`0.210.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.

   a. Copiez le contenu suivant dans un fichier nommé *`efa-cluster.yaml`*. Remplacez les *`example values`* par vos propres valeurs. Vous pouvez remplacer *`p5.48xlarge`* par une instance différente. Dans ce cas, assurez-vous que les valeurs de `availabilityZones` sont des zones de disponibilité renvoyées pour le type d'instance à l'étape 1.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

b. Créez un groupe de nœuds gérés dans un cluster existant.

```
eksctl create nodegroup -f efa-cluster.yaml
```

Si vous n'avez pas de cluster existant, vous pouvez exécuter la commande suivante pour créer un cluster et le groupe de nœuds.

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> Comme le type d'instance utilisé dans cet exemple l'est GPUs, le plug-in pour appareil NVIDIA Kubernetes est `eksctl` automatiquement installé sur chaque instance lorsque vous utilisez Amazon Linux 2. Cela n'est pas nécessaire pour Bottlerocket, car le plug-in pour appareil NVIDIA est intégré à la variante EKS NVIDIA de Bottlerocket. Lorsque `efaEnabled` ce paramètre est défini sur `true` dans la configuration du groupe de nœuds, le plug-in de périphérique EFA `eksctl` sera également automatiquement déployé sur les nœuds.

Utiliser Bottlerocket avec EFA

Les versions 1.28.0 et ultérieures de l'AMI Bottlerocket incluent le support officiel pour EFA. Pour utiliser Bottlerocket pour les nœuds compatibles EFA, spécifiez-le dans votre configuration. `amiFamily: Bottlerocket` Si vous devez utiliser un ID d'AMI personnalisé, vous devez utiliser le standard `nodeGroups` au lieu de`managedNodeGroups`.

Voici un exemple de configuration :

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-bottlerocket-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true
```

```
availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-bottlerocket-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
      settings:
        kernel:
          sysctl:
            "vm.nr_hugepages": "3000"  # Configures 3000 * 2Mi = 6000Mi hugepages
```

Le paramètre `vm.nr_hugepages` sysctl ci-dessus configure le nombre de pages énormes de 2 Mo. Dans cet exemple, 3000 signifie 3 000* 2 Mi = 6 000 Mi de pages immenses.

Vérifier l'installation du plug-in de l'appareil EFA

Lorsque vous créez un groupe de nœuds avec`efaEnabled: true`, déploie `eksctl` automatiquement le plug-in d'appareil EFA Kubernetes pour vous. Vous pouvez vérifier que le plug-in de l'appareil est installé et fonctionne correctement :

1. Vérifiez le DaemonSet statut :

   ```
   kubectl get daemonsets -n kube-system
   ```

   Exemple de sortie :

   ```
   NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE
    AVAILABLE   NODE SELECTOR   AGE
   aws-efa-k8s-device-plugin-daemonset   2         2         2       2            2
         <none>          6m16s
   ...
   ```

Ici, le plug-in de l'appareil EFA DaemonSet s'exécute sur deux nœuds. Les deux sont prêts et disponibles.

2. Ensuite, vérifiez les pods créés par DaemonSet :

```
kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
```

Exemple de sortie :

```
NAME                                          READY   STATUS    RESTARTS   AGE
aws-efa-k8s-device-plugin-daemonset-d68bs   1/1     Running   0          6m16s
aws-efa-k8s-device-plugin-daemonset-w4l8t   1/1     Running   0          6m16s
```

Les pods du plug-in de l'appareil EFA sont en cours d'exécution, ce qui confirme que le plug-in est correctement déployé et opérationnel.

3. Vérifiez l'enregistrement des ressources :

Vous pouvez confirmer que la `vpc.amazonaws.com/efa` ressource est enregistrée auprès du kubelet en décrivant les nœuds :

```
kubectl describe nodes
```

Si la ressource EFA est correctement enregistrée, vous la verrez répertoriée sous la capacité et les ressources allouables du nœud. Par exemple :

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

Cette sortie confirme que le nœud reconnaît la ressource EFA et la rend disponible pour les pods qui en font la demande.

(Facultatif) Testez les performances de l'EFA

Nous vous recommandons de tester la configuration EFA. Vous pouvez utiliser les tests NCCL dans le `aws-samples/awsome-distributed-training` référentiel sur. GitHub Les tests NCCL évaluent les performances du réseau à l'aide de la bibliothèque de communication collective Nvidia. Les étapes suivantes soumettent les tests NCCL sur Amazon EKS.

1. Déployez l'opérateur MPI Kubeflow :

   Pour les tests NCCL, vous pouvez appliquer l'opérateur MPI Kubeflow. L'opérateur MPI facilite l'exécution d'un entraînement distribué de style AllReduce sur Kubernetes. Pour plus d'informations, consultez la section Opérateur MPI activé GitHub.

2. Exécutez le test de performance NCCL multi-nœuds pour vérifier le RDMA/EFA : GPUDirect

   Pour vérifier les performances du NCCL avec GPUDirect RDMA sur EFA, exécutez le test de performance NCCL standard. Pour plus d'informations, consultez le dépôt officiel des tests NCCL sur. GitHub

   Procédez comme indiqué ci-dessous pour exécuter un test de performance NCCL à deux nœuds. Dans l'exemple de tâche de test NCCL, chaque travailleur demande huit GPUs, 5 210 Mohugepages-2Mi, quatre et 8 000 Mo de mémoire EFAs, ce qui signifie que chaque travailleur consomme toutes les ressources d'une instance. `p5.48xlarge`

   a. Créez le MPIJob manifeste :

      Copiez ce qui suit dans un fichier nommé `nccl-tests.yaml` :

```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccl-tests
spec:
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
```

```
          restartPolicy: OnFailure
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: test-nccl-launcher
            env:
             - name: PATH
               value: $PATH:/opt/amazon/efa/bin:/usr/bin
             - name: LD_LIBRARY_PATH
               value: /opt/amazon/openmpi/lib:/opt/nccl/build/lib:/opt/amazon/efa/
lib:/opt/aws-ofi-nccl/install/lib:/usr/local/nvidia/lib:$LD_LIBRARY_PATH
             - name: NCCL_DEBUG
               value: INFO
             - name: NCCL_BUFFSIZE
               value: '8388608'
             - name: NCCL_P2P_NET_CHUNKSIZE
               value: '524288'
             - name: NCCL_TUNER_PLUGIN
               value: /opt/aws-ofi-nccl/install/lib/libnccl-ofi-tuner.so
            command:
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none
            - -x
            - PATH
            - -x
            - LD_LIBRARY_PATH
            - -x
            - NCCL_DEBUG=INFO
            - -x
            - NCCL_BUFFSIZE
            - -x
            - NCCL_P2P_NET_CHUNKSIZE
            - -x
            - NCCL_TUNER_PLUGIN
            - --mca
            - pml
            - ^cm,ucx
```

```
            - --mca
            - btl
            - tcp,self
            - --mca
            - btl_tcp_if_exclude
            - lo,docker0,veth_def_agent
            - /opt/nccl-tests/build/all_reduce_perf
            - -b
            - "8"
            - -e
            - "16G"
            - -f
            - "2"
            - -g
            - "1"
            - -c
            - "1"
            - -n
            - "100"
    Worker:
      replicas: 2
      template:
        spec:
          nodeSelector:
            node.kubernetes.io/instance-type: "p5.48xlarge"
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: nccl-tests-worker
            volumeMounts:
            - name: shmem
              mountPath: /dev/shm
            resources:
              limits:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
              requests:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
          volumes:
```

```
          - name: shmem
            hostPath:
              path: /dev/shm
```

b. Appliquez les tests NCCL : MPIJob

   `MPIJob`Soumettez-les en appliquant le manifeste. Cela créera deux EC2 instances `p5.48xlarge` Amazon.

```
kubectl apply -f nccl-tests.yaml
```

   L'exemple qui suit illustre un résultat.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Vérifiez que le job a démarré :

   Visualisez vos pods de course.

```
kubectl get pods
```

   L'exemple qui suit illustre un résultat.

```
NAME                            READY   STATUS     RESTARTS   AGE
nccl-tests-launcher-nbql9    0/1     Init:0/1   0          2m49s
nccl-tests-worker-0          1/1     Running    0          2m49s
nccl-tests-worker-1          1/1     Running    0          2m49s
```

   L'opérateur MPI crée un module de lancement et 2 pods de travail (un sur chaque nœud).

d. Vérifiez que la tâche s'exécute correctement avec les journaux :

   Consultez le journal du `nccl-tests-launcher` Pod. Remplacez *`nbql9`* par la valeur de votre sortie.

```
kubectl logs -f nccl-tests-launcher-nbql9
```

Si le test est réussi, vous pouvez déployer vos applications qui utilisent la bibliothèque de communication collective Nvidia.

Utiliser des instances AWS Inferentia avec Amazon EKS pour le Machine Learning

Cette rubrique décrit comment créer un cluster Amazon EKS avec des nœuds exécutant des instances Amazon EC2 Inf1 et (éventuellement) déployer un exemple d'application. Les instances Amazon EC2 Inf1 sont alimentées par des puces AWS Inferentia, conçues sur mesure AWS pour fournir des performances élevées et une inférence à moindre coût dans le cloud. Les modèles d'apprentissage automatique sont déployés sur des conteneurs à l'aide de AWS Neuron, un kit de développement logiciel (SDK) spécialisé composé d'un compilateur, d'un environnement d'exécution et d'outils de profilage qui optimisent les performances d'inférence par apprentissage automatique des puces Inferentia. AWS Neuron prend en charge les frameworks d'apprentissage automatique populaires tels que TensorFlow PyTorch, et MXNet.

> **Note**
>
> La logique du dispositif neuronal IDs doit être contiguë. Si un pod demandant plusieurs appareils Neuron est planifié sur un type d'`inf1.24xlarge`instance `inf1.6xlarge` ou (qui possède plusieurs appareils Neuron), ce pod ne démarrera pas si le planificateur Kubernetes sélectionne un appareil non contigu. IDs Pour plus d'informations, consultez la section Le périphérique logique IDs doit être contigu. GitHub

Prérequis

- Installez `eksctl` sur votre ordinateur. Si ce n'est pas le cas, consultez la section Installation dans la `eksctl` documentation.
- Installez `kubectl` sur votre ordinateur. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".
- (Facultatif) Installez `python3` sur votre ordinateur. Si vous ne l'avez pas installé, consultez les téléchargements de Python pour les instructions d'installation.

Créer un cluster

1. Créez un cluster avec des nœuds d' EC2 instance Amazon Inf1. Vous pouvez remplacer *`inf1.2xlarge`* par n'importe quel type d'instance Inf1. L'`eksctl`utilitaire détecte que vous lancez

un groupe de nœuds avec un type d'`Inf1`instance et démarrera vos nœuds à l'aide de l'un des systèmes accélérés optimisés pour Amazon EKS d'Amazon Linux AMIs.

> **Note**
>
> Vous ne pouvez pas utiliser de rôles IAM pour les comptes de service avec TensorFlow Serving.

```
eksctl create cluster \
    --name inferentia \
    --region region-code \
    --nodegroup-name ng-inf1 \
    --node-type inf1.2xlarge \
    --nodes 2 \
    --nodes-min 1 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key your-key \
    --with-oidc
```

> **Note**
>
> Notez la valeur de la ligne suivante de la sortie. Il est utilisé dans une étape ultérieure (facultative).

```
[9]  adding identity "arn:aws: iam::111122223333:role/eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

Lorsque vous lancez un groupe de nœuds avec des `Inf1` instances, le plug-in d'appareil AWS Neuron Kubernetes est `eksctl` automatiquement installé. Ce plug-in annonce les appareils Neuron comme une ressource système au planificateur Kubernetes, qui peut être demandé par un conteneur. Outre les stratégies IAM de nœud Amazon EKS par défaut, la stratégie d'accès en lecture seule Amazon S3 est ajoutée afin que l'exemple d'application, traité dans une étape ultérieure, puisse charger un modèle formé à partir d'Amazon S3.

2. Assurez-vous que tous les pods ont démarré correctement.

```
kubectl get pods -n kube-system
```

Sortie abrégée :

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(Facultatif) Déployez une image d'application TensorFlow Serving

Un modèle formé doit être compilé sur une cible Inferentia avant de pouvoir être déployé sur des instances Inferentia. Pour continuer, vous aurez besoin d'un TensorFlow modèle optimisé pour Neuron enregistré dans Amazon S3. Si vous n'en avez pas encore SavedModel, veuillez suivre le didacticiel pour créer un modèle ResNet 50 compatible avec Neuron et télécharger le résultat SavedModel sur S3. ResNet-50 est un modèle d'apprentissage automatique populaire utilisé pour les tâches de reconnaissance d'images. Pour plus d'informations sur la compilation de modèles de neurones, consultez The AWS Inferentia Chip with DLAMI dans le Deep Learning Developer Guide. AWS AMIs

L'exemple de manifeste de déploiement gère un conteneur de service d'inférence prédéfini TensorFlow fourni par AWS Deep Learning Containers. À l'intérieur du conteneur se trouvent le AWS Neuron Runtime et l'application TensorFlow Serving. Une liste complète des Deep Learning Containers prédéfinis optimisés pour Neuron est disponible GitHub sous Images disponibles. Au démarrage, le DLC récupérera votre modèle sur Amazon S3, lancera Neuron TensorFlow Serving avec le modèle enregistré et attendra les demandes de prédiction.

Le nombre d'appareils Neuron alloués à votre application de service peut être ajusté en changeant la ressource `aws.amazon.com/neuron` dans le yaml de déploiement. Veuillez noter que la communication entre TensorFlow Serving et le runtime Neuron se fait via GRPC, ce qui nécessite de transmettre la `IPC_LOCK` capacité au conteneur.

1. Ajoutez la politique `AmazonS3ReadOnlyAccess` IAM au rôle d'instance de nœud créé à l'étape 1 de la section Créer un cluster. Ceci est nécessaire pour que l'application exemple puisse charger un modèle formé à partir de Amazon S3.

   ```
   aws iam attach-role-policy \
   ```

```
    --policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. Créez un fichier nommé `rn50_deployment.yaml` avec les contenus suivants. Mettez à jour le code régional et le chemin du modèle pour correspondre aux paramètres souhaités. Le nom du modèle est utilisé à des fins d'identification lorsqu'un client fait une demande au TensorFlow serveur. Cet exemple utilise un nom de modèle correspondant à un exemple de ResNet 50 scripts client qui sera utilisé ultérieurement pour envoyer des demandes de prédiction.

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --region
 us-west-2
```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://${your-bucket-of-models}/resnet50_neuron/
```

```
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. Déployez le modèle.

```
kubectl apply -f rn50_deployment.yaml
```

4. Créez un fichier nommé `rn50_service.yaml` avec les contenus suivants. Les ports HTTP et gRPC sont ouverts pour accepter les demandes de prédiction.

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
```

```
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. Créez un service Kubernetes pour votre application TensorFlow Model Serving.

```
kubectl apply -f rn50_service.yaml
```

(Facultatif) Faites des prédictions par rapport à votre TensorFlow service de service

1. Pour tester localement, transférez le port gRPC au service `eks-neuron-test`.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Créez un script Python appelé `tensorflow-model-server-infer.py` avec le contenu suivant. Ce script exécute l'inférence via gRPC, qui est une infrastructure de service.

```
import numpy as np
   import grpc
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions

   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
       img_file = tf.keras.utils.get_file(
           "./kitten_small.jpg",
           "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
       img = image.load_img(img_file, target_size=(224, 224))
```

```
        img_array = preprocess_input(image.img_to_array(img)[None, ...])
        request = predict_pb2.PredictRequest()
        request.model_spec.name = 'resnet50_inf1'
        request.inputs['input'].CopyFrom(
            tf.make_tensor_proto(img_array, shape=img_array.shape))
        result = stub.Predict(request)
        prediction = tf.make_ndarray(result.outputs['output'])
        print(decode_predictions(prediction))
```

3. Exécutez le script pour soumettre des prédictions à votre service.

```
python3 tensorflow-model-server-infer.py
```

L'exemple qui suit illustre un résultat.

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

Gérez les ressources informatiques pour les AI/ML charges de travail sur Amazon EKS

Cette section est conçue pour vous aider à gérer les ressources de calcul pour les charges de travail d'apprentissage automatique dans Amazon Elastic Kubernetes Service (EKS). Vous trouverez des informations sur la réservation à GPUs l'aide de blocs de capacité pour les groupes de nœuds gérés et les nœuds autogérés, notamment les conditions préalables, la configuration du modèle de lancement, les configurations de dimensionnement, la préparation de la charge de travail, ainsi que les principales considérations relatives à la gestion des cycles de vie des réservations et à la résiliation progressive des nœuds.

Rubriques

- Création d'un groupe de nœuds géré avec Capacity Blocks for ML
- Créez des nœuds autogérés avec Capacity Blocks for ML

Création d'un groupe de nœuds géré avec Capacity Blocks for ML

Les blocs de capacité pour le Machine Learning (ML) vous permettent de réserver des instances GPU à une date ultérieure pour prendre en charge vos charges de travail de ML de courte durée. Pour plus d'informations, consultez Capacity Blocks for ML dans le guide de EC2 l'utilisateur Amazon pour les instances Linux.

Considérations

Important

- Les blocs de capacité ne sont disponibles que pour certains types d' EC2 instances Amazon et certaines AWS régions. Pour plus d'informations sur la compatibilité, consultez les conditions préalables à l'utilisation des blocs de capacité dans le guide de EC2 l'utilisateur Amazon pour les instances Linux.
- Pour plus d'informations, consultez la section Utiliser les blocs de capacité pour les charges de travail d'apprentissage automatique dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
- Les groupes de nœuds gérés avec des blocs de capacité ne peuvent être créés qu'avec des modèles de lancement personnalisés.
- Lorsque vous mettez à niveau des groupes de nœuds gérés avec des blocs de capacité, assurez-vous que la taille souhaitée du groupe de nœuds est définie sur`0`.

Créez un groupe de nœuds gérés avec Amazon EC2 Capacity Blocks

Vous pouvez utiliser les blocs de capacité avec les groupes de nœuds gérés par Amazon EKS pour provisionner et dimensionner des nœuds de travail accélérés par GPU. Les exemples de AWS CloudFormation modèles suivants ne couvrent pas tous les aspects nécessaires à un cluster de production. Généralement, vous souhaiteriez également qu'un script d'amorçage relie le nœud au cluster et spécifie une AMI accélérée Amazon EKS. Pour de plus amples informations, veuillez consulter the section called “Création”.

1. Créez un modèle de lancement adapté à vos charges de travail et compatible avec les groupes de nœuds gérés par Amazon EKS. Pour de plus amples informations, veuillez consulter the section called “Modèles de lancement”.

Outre les exigences des procédures ci-dessus, assurez-vous qu'elles `LaunchTemplateData` incluent les éléments suivants :

- `InstanceMarketOptions` avec `MarketType` défini sur `"capacity-block"`
- `CapacityReservationSpecification: CapacityReservationTarget`avec `CapacityReservationId` défini sur le bloc de capacité (par exemple :`cr-`*`02168da1478b509e0`*)
- `InstanceType`défini sur un type d'instance qui prend en charge les blocs de capacité (par exemple :*`p5.48xlarge`*)

  Voici un extrait d'un CloudFormation modèle qui crée un modèle de lancement ciblant un bloc de capacité. Pour créer un groupe de nœuds géré par AMI personnalisé, vous pouvez également ajouter `ImageId` des `UserData` paramètres.

```
NodeLaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      InstanceType: p5.48xlarge
```

2. Utilisez le modèle de lancement pour créer un groupe de nœuds gérés.

   Voici un exemple de commande de création de groupe de nœuds pour Capacity Blocks. *`example-values`*Remplacez-les par ceux applicables à votre cluster.

   Lors de la création du groupe de nœuds géré par Capacity Block, procédez comme suit :

   - Définissez `capacity-type` sur `"CAPACITY_BLOCK"`. Si le type de capacité n'est pas défini sur `"CAPACITY_BLOCK"` ou si l'une des autres valeurs de modèle de lancement requises ci-dessus est manquante, la demande de création sera rejetée.
   - Lorsque vous spécifiez `subnets` dans la demande de création, assurez-vous de ne spécifier que le sous-réseau dans la même zone de disponibilité que la réservation de capacité.
   - Si vous spécifiez une valeur différente de zéro `desiredSize` dans la demande de création, Amazon EKS respectera cette valeur lors de la création du groupe Auto Scaling (ASG).

Toutefois, si la demande de création est effectuée avant que la réservation de capacité ne soit active, l'ASG ne sera pas en mesure de lancer les EC2 instances Amazon tant que celle-ci ne sera pas active. Par conséquent, les activités de dimensionnement d'ASG comporteront des erreurs de lancement. Chaque fois que la réservation devient active, le lancement des instances aboutira et l'ASG sera étendu au niveau `desiredSize` mentionné au moment de la création.

```
aws eks create-nodegroup \
    --cluster-name my-cluster \
    --nodegroup-name my-mng \
    --node-role node-role-arn \
    --region region-code \
    --subnets subnet-id \
    --scaling-config minSize=node-group-min-size,maxSize=node-group-max-
size,desiredSize=node-group-desired-size \
    --ami-type "AL2023_x86_64_NVIDIA" \
    --capacity-type "CAPACITY_BLOCK" \
    --launch-template id="lt-id",version=1
```

3. Assurez-vous que les nœuds se rejoignent après la mise à l'échelle. Les clusters Amazon EKS utilisant des groupes de nœuds gérés avec des blocs de capacité n'effectuent aucune validation indiquant que les instances lancées rejoignent le cluster et s'enregistrent auprès de celui-ci.
4. Si vous l'avez défini `desiredSize 0` au moment de la création, vous disposez de différentes options pour augmenter le groupe de nœuds lorsque la réservation de capacité devient active :
   - Créez une politique de dimensionnement planifiée pour l'ASG qui s'aligne sur l'heure de début de la réservation du Capacity Block. Pour plus d'informations, consultez la section Scheduled Scaling for Amazon EC2 Auto Scaling dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
   - Utilisez la console Amazon EKS ou `eks update-nodegroup-config` pour mettre à jour la configuration de dimensionnement et définir la taille souhaitée du groupe de nœuds.
   - Utilisez le Kubernetes Cluster Autoscaler. Pour plus d'informations, consultez Cluster Autoscaler activé. AWS
5. Le groupe de nœuds est désormais prêt pour les charges de travail et les pods à planifier.
6. Pour que vos pods soient correctement vidés avant la fin de la réservation, Amazon EKS utilise une politique de dimensionnement planifiée pour réduire la taille du groupe de nœuds à`0`. Cette mise à l'échelle planifiée sera définie avec un nom intituléAmazon `EKS Node Group Capacity Scaledown Before Reservation End`. Nous vous recommandons de ne pas modifier ou supprimer cette action.

Amazon EC2 commence à fermer les instances 30 minutes avant la fin de la réservation. Par conséquent, Amazon EKS mettra en place une réduction planifiée du groupe de nœuds 40 minutes avant la fin de leur réservation afin d'expulser les Pods de manière sûre et élégante.

Créez des nœuds autogérés avec Capacity Blocks for ML

Les blocs de capacité pour le Machine Learning (ML) vous permettent de réserver des instances GPU à une date ultérieure pour prendre en charge vos charges de travail de ML de courte durée. Pour plus d'informations, consultez Capacity Blocks for ML dans le guide de EC2 l'utilisateur Amazon pour les instances Linux.

Considérations

Important

- Les blocs de capacité ne sont disponibles que pour certains types d' EC2 instances Amazon et certaines AWS régions. Pour plus d'informations sur la compatibilité, consultez les conditions préalables à l'utilisation des blocs de capacité dans le guide de EC2 l'utilisateur Amazon pour les instances Linux.
- Si vous créez un groupe de nœuds autogéré avant que la réservation de capacité ne soit active, définissez la capacité souhaitée sur`0`.
- Afin de disposer de suffisamment de temps pour vidanger le(s) nœud(s) de manière gracieuse, nous vous conseillons de programmer une mise à l'échelle jusqu'à zéro plus de 30 minutes avant l'heure de fin de la réserve du bloc de capacité.
- Pour que vos pods soient correctement vidés, nous vous recommandons de configurer le AWS Node Termination Handler comme expliqué dans les étapes d'exemple.

Utiliser des blocs de capacité avec des nœuds autogérés

Vous pouvez utiliser les blocs de capacité avec Amazon EKS pour l'approvisionnement et la mise à l'échelle de vos nœuds autogérés. Les étapes suivantes vous donnent un aperçu général de l'exemple. Les exemples AWS CloudFormation de modèles ne couvrent pas tous les aspects nécessaires à une charge de travail de production. Généralement, vous souhaitez également qu'un script d'amorçage relie le nœud au cluster, spécifie une AMI accélérée Amazon EKS et un profil

d'instance approprié pour rejoindre le cluster. Pour de plus amples informations, veuillez consulter the section called "Amazon Linux".

1. Créez un modèle de lancement applicable à votre charge de travail. Pour plus d'informations, consultez la section Utiliser les blocs de capacité pour les charges de travail d'apprentissage automatique dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.

   Assurez-vous qu'il `LaunchTemplateData` inclut les éléments suivants :

   - `InstanceMarketOptions` avec `MarketType` défini sur `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget`avec `CapacityReservationId` défini sur le bloc de capacité (par exemple :`cr-`*`02168da1478b509e0`*)
   - `IamInstanceProfile`avec le `Arn` set correspondant *`iam-instance-profile-arn`*
   - `ImageId`réglé sur la valeur applicable *`image-id`*
   - `InstanceType`défini sur un type d'instance qui prend en charge les blocs de capacité (par exemple :*`p5.48xlarge`*)
   - `SecurityGroupIds`défini sur la valeur applicable IDs (par exemple :*`sg-05b1d815d1EXAMPLE`*)
   - `UserData`défini sur celui applicable *`user-data`* à votre groupe de nœuds autogéré

   Voici un extrait d'un CloudFormation modèle qui crée un modèle de lancement ciblant un bloc de capacité.

```
NodeLaunchTemplate:
  Type: "aws::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      IamInstanceProfile:
        Arn: iam-instance-profile-arn
      ImageId: image-id
      InstanceType: p5.48xlarge
      KeyName: key-name
      SecurityGroupIds:
      - sg-05b1d815d1EXAMPLE
```

```
      UserData: user-data
```

Vous devez transmettre le sous-réseau de la zone de disponibilité dans laquelle la réservation est effectuée, car les blocs de capacité sont répartis par zone.

2. Utilisez le modèle de lancement pour créer un groupe de nœuds autogéré. Si vous effectuez cette opération avant que la réservation de capacité ne soit active, définissez la capacité souhaitée sur`0`. Lorsque vous créez le groupe de nœuds, assurez-vous que vous ne spécifiez que le sous-réseau correspondant à la zone de disponibilité dans laquelle la capacité est réservée.

   Voici un exemple de CloudFormation modèle auquel vous pouvez faire référence lors de la création d'un modèle applicable à votre charge de travail. Cet exemple permet d'obtenir le `LaunchTemplateId` et `Version` de la `AWS::Amazon EC2::LaunchTemplate` ressource indiquée à l'étape précédente. Il obtient également les valeurs pour `DesiredCapacity`, `MaxSize`, `MinSize`, et `VPCZoneIdentifier` qui sont déclarées ailleurs dans le même modèle.

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
        PropagateAtLaunch: true
        Value: owned
```

3. Une fois que le groupe de nœuds a été créé avec succès, assurez-vous d'enregistrer le `NodeInstanceRole` pour le groupe de nœuds qui a été créé. Vous en avez besoin pour vous assurer que lorsque le groupe de nœuds est dimensionné, les nouveaux nœuds rejoignent le cluster et que Kubernetes est capable de reconnaître les nœuds. Pour plus d'informations, consultez les AWS Management Console instructions de la section Création de nœuds Amazon Linux autogérés.

4. Nous vous recommandons de créer une politique de mise à l'échelle programmée pour le groupe Auto Scaling qui s'aligne sur les heures de réserve du bloc de capacité. Pour plus d'informations, consultez la section Scheduled Scaling for Amazon EC2 Auto Scaling dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.

   Vous pouvez utiliser toutes les instances que vous avez réservées jusqu'à 30 minutes avant l'heure de fin du bloc de capacité. Les instances encore en cours d'exécution à ce moment-là commenceront à prendre fin. Afin de disposer de suffisamment de temps pour vidanger le(s) nœud(s) de manière gracieuse, nous vous conseillons de programmer une mise à l'échelle jusqu'à zéro plus de 30 minutes avant l'heure de fin de la réserve du bloc de capacité.

   Si vous souhaitez plutôt augmenter manuellement chaque fois que la réservation de capacité est effectuée`Active`, vous devez mettre à jour la capacité souhaitée du groupe Auto Scaling au début de la réservation du Capacity Block. Ensuite, vous devrez également réduire manuellement la capacité plus de 30 minutes avant l'heure de fin de la réserve du bloc de capacité.
5. Le groupe de nœuds est désormais prêt pour les charges de travail et les pods à planifier.
6. Pour que vos pods soient correctement vidés, nous vous recommandons de configurer AWS Node Termination Handler. Ce gestionnaire sera en mesure de surveiller les événements du cycle de vie « ASG Scale-in » liés à Amazon EC2 Auto Scaling en utilisant le plan de contrôle Kubernetes EventBridge et de permettre au plan de contrôle Kubernetes de prendre les mesures nécessaires avant que l'instance ne soit indisponible. Sinon, vos pods et objets Kubernetes resteront bloqués dans un état d'attente. Pour plus d'informations, consultez AWS Node Termination Handler activé GitHub.

   Si vous ne configurez pas de gestionnaire de terminaison de nœuds, nous vous recommandons de commencer à vider vos pods manuellement avant d'atteindre le délai de 30 minutes afin qu'ils aient suffisamment de temps pour être correctement vidés.

Recettes pour optimiser votre cluster Amazon EKS en fonction des charges de AI/ML travail

Cette section est conçue pour fournir de petites recettes pour optimiser votre cluster Amazon EKS, en particulier pour les AI/ML charges de travail impliquant du matériel spécialisé. Vous trouverez des conseils pour empêcher la planification des pods sur des nœuds spécifiques en alourdissant les groupes de nœuds gérés, notamment les conditions préalables, les step-by-step procédures et les considérations relatives au déploiement.

Rubriques

- Recette : empêcher la planification des pods sur des nœuds spécifiques

Recette : empêcher la planification des pods sur des nœuds spécifiques

Présentation

Les nœuds dotés de processeurs spécialisés, par exemple GPUs, peuvent être plus coûteux à exécuter que les nœuds sur des machines standard. Pour protéger ces nœuds des charges de travail ne nécessitant pas de matériel spécial, vous pouvez utiliser Kubernetes Taints. Des taches marquent les nœuds pour repousser les pods qui ne présentent pas les mêmes tolérances, garantissant ainsi que seules les charges de travail compatibles sont planifiées. Pour plus d'informations, consultez Taints and Tolerations dans la documentation de Kubernetes.

Les altérations de nœuds Kubernetes peuvent être appliquées à des groupes de nœuds gérés nouveaux et existants à l'aide de l'API Amazon EKS ou AWS Management Console via celle-ci. Cette recette explique comment appliquer des altérations aux groupes de nœuds gérés par Amazon EKS à l'aide de la AWS CLI. Pour plus d'informations sur la création d'un groupe de nœuds teinté à l'aide du AWS Management Console, voirthe section called "Création".

Prérequis

- Un cluster Amazon EKS existant.
- AWS CLI installée et configurée avec les autorisations appropriées.

Étapes

Étape 1 : créer un groupe de nœuds présentant des taches

Utilisez la `aws eks create-nodegroup` commande pour créer un nouveau groupe de nœuds gérés présentant des taches. Cet exemple applique une teinte avec une clé`dedicated`, une valeur `gpuGroup` et un effet. `NO_SCHEDULE`

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
```

```
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws: iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

Pour plus d'informations et des exemples, consultez taint dans la documentation de référence de Kubernetes.

Étape 2 : Mettre à jour les taches sur un groupe de nœuds existant

Utilisez la commande AWS eks update-nodegroup-config AWS CLI pour ajouter, supprimer ou remplacer des taches pour les groupes de nœuds gérés.

```
aws eks update-nodegroup-config
  --cluster-name my-cluster
  --nodegroup-name node-taints-example
  --taints 'removeTaints=[{key=dedicated,value=gpuGroup,effect=NO_SCHEDULE}]'
```

Remarques

- Les rejets peuvent être mis à jour après avoir créé le groupe de nœuds à l'aide de l'API `UpdateNodegroupConfig`.
- La clé de rejet doit commencer par une lettre ou un chiffre. Il peut contenir des lettres, des chiffres, des traits d'union (`-`), des points (`.`) et des traits de soulignement (_). Il peut comporter jusqu'à 63 caractères.
- Éventuellement, la clé de rejet peut commencer par un préfixe de sous-domaine DNS et un simple `/`. Si elle commence par un préfixe de sous-domaine DNS, elle peut comporter 253 caractères.
- La valeur est facultative et doit commencer par une lettre ou un chiffre. Il peut contenir des lettres, des chiffres, des traits d'union (`-`), des points (`.`) et des traits de soulignement (_). Il peut comporter jusqu'à 63 caractères.

- Lorsque vous utilisez Kubernetes directement ou le AWS Management Console, l'effet de teinte doit êtreNoSchedule, ou. PreferNoSchedule NoExecute Toutefois, lors de l'utilisation de la AWS CLI ou de l'API, l'effet d'altération doit être NO_SCHEDULEPREFER_NO_SCHEDULE, ouNO_EXECUTE.
- Un maximum de 50 rejets est autorisé pour un groupe de nœuds.
- Si les souillures créées à l'aide d'un groupe de nœuds géré sont supprimées manuellement d'un nœud, Amazon EKS ne les réajoute pas au nœud. Ceci est vrai même si les rejets sont spécifiés dans la configuration du groupe de nœuds gérés.

Ressources pour démarrer AI/ML sur Amazon EKS

Pour vous lancer dans le Machine Learning sur EKS, commencez par choisir parmi ces modèles prescriptifs afin de préparer rapidement un cluster EKS ainsi que des logiciels et du matériel ML à exécuter des charges de travail ML.

Ateliers

Atelier EKS sur l'IA générative sur Amazon

Découvrez comment démarrer avec les applications LLM (Large Language Model) et l'inférence sur Amazon EKS. Découvrez comment déployer et gérer des charges de travail LLM de niveau production. Dans le cadre d'ateliers pratiques, vous découvrirez comment tirer parti d'Amazon EKS ainsi que des AWS services et des outils open source pour créer des solutions LLM robustes. L'environnement de l'atelier fournit toutes les infrastructures et tous les outils nécessaires, vous permettant de vous concentrer sur l'apprentissage et la mise en œuvre.

IA générative sur Amazon EKS à l'aide de Neuron

Découvrez comment démarrer avec les applications LLM (Large Language Model) et l'inférence sur Amazon EKS. Découvrez comment déployer et gérer des charges de travail LLM de niveau production, implémenter des modèles RAG avancés avec des bases de données vectorielles et créer des applications LLM basées sur des données à l'aide de frameworks open source. Dans le cadre d'ateliers pratiques, vous découvrirez comment tirer parti d'Amazon EKS ainsi que des AWS services et des outils open source pour créer des solutions LLM robustes. L'environnement de l'atelier fournit toutes les infrastructures et tous les outils nécessaires, vous permettant de vous concentrer sur l'apprentissage et la mise en œuvre.

Bonnes pratiques

Les AI/ML rubriques spécifiques du guide des meilleures pratiques Amazon EKS fournissent des recommandations détaillées dans les domaines suivants pour optimiser vos AI/ML charges de travail sur Amazon EKS.

Calcul AI/ML et mise à l'échelle automatique

Cette section décrit les meilleures pratiques pour optimiser le AI/ML calcul et le dimensionnement automatique dans Amazon EKS, en se concentrant sur la gestion des ressources GPU, la résilience des nœuds et le dimensionnement des applications. Il propose des stratégies telles que la planification des charges de travail avec des étiquettes et une affinité de nœuds bien connues, l'utilisation de blocs de capacité ML ou de réservations de capacité à la demande, et la mise en œuvre de contrôles de santé des nœuds à l'aide d'outils tels que l'agent de surveillance des nœuds EKS.

Mise en réseau AI/ML

Cette section décrit les meilleures pratiques pour optimiser le AI/ML réseau dans Amazon EKS afin d'améliorer les performances et l'évolutivité, notamment des stratégies telles que la sélection d'instances dotées d'une bande passante réseau plus élevée ou d'Elastic Fabric Adapter (EFA) pour la formation distribuée, l'installation d'outils tels que MPI et NCCL, et l'activation de la délégation de préfixes pour augmenter le nombre d'adresses IP et améliorer les temps de lancement des pods.

Sécurité de l'IA et du ML

Cette section se concentre sur la sécurisation du stockage des données et la garantie de la conformité des AI/ML charges de travail sur Amazon EKS, notamment des pratiques telles que l'utilisation d'Amazon S3 avec AWS Key Management Service (KMS) pour le chiffrement côté serveur (SSE-KMS), la configuration de compartiments avec des clés KMS régionales et des clés de compartiment S3 pour réduire les coûts, l'octroi d'autorisations IAM pour les actions KMS telles que le déchiffrement des pods EKS et l'audit avec des journaux. AWS CloudTrail

Stockage AI/ML

Cette section décrit les meilleures pratiques pour optimiser le stockage dans les AI/ML charges de travail sur Amazon EKS, notamment le déploiement de modèles utilisant des pilotes CSI pour monter des services tels que S3, FSx pour Lustre ou EFS sous forme de volumes persistants, la sélection

du stockage en fonction des besoins de charge de travail (par exemple, FSx pour Lustre pour la formation distribuée avec des options telles que Scratch-SSD ou Persistent-SSD) et l'activation de fonctionnalités telles que la compression et le découpage des données.

Observabilité de l'AI/ML

Cette section se concentre sur la surveillance et l'optimisation de l'utilisation du GPU pour les AI/ML charges de travail sur Amazon EKS afin d'améliorer l'efficacité et de réduire les coûts, notamment des stratégies telles que le ciblage d'une utilisation élevée du GPU avec des outils tels que CloudWatch Container Insights et l'exportateur DCGM de NVIDIA intégré à Prometheus et Grafana, ainsi que des indicateurs que nous vous recommandons d'analyser pour vos charges de travail. AI/ML

Performances d'IA et d'apprentissage automatique

Cette section se concentre sur l'amélioration de la mise à l'échelle et des performances des applications pour les AI/ML charges de travail sur Amazon EKS grâce à la gestion des images de conteneur et à l'optimisation du démarrage, y compris des pratiques telles que l'utilisation de petites images de base légères ou de AWS Deep Learning Containers avec des versions en plusieurs étapes, le préchargement d'images via des instantanés EBS ou le pré-extraction dans le cache d'exécution à l'aide de déploiements. DaemonSets

Architectures de référence

Explorez ces GitHub référentiels pour trouver des architectures de référence, des exemples de code et des utilitaires permettant de mettre en œuvre une formation distribuée et une inférence pour les charges de AI/ML travail sur Amazon EKS et d'autres services. AWS

AWSome Formation distribuée

Ce référentiel propose un ensemble de bonnes pratiques, d'architectures de référence, d'exemples de formation de modèles et d'utilitaires pour l'entraînement de grands modèles AWS. Il prend en charge la formation distribuée avec Amazon EKS, notamment des CloudFormation modèles pour les clusters EKS, des versions personnalisées d'AMI et de conteneurs, des scénarios de test pour des frameworks tels que PyTorch (DDP/FSDP, MegatronLM NeMo) et JAX, ainsi que des outils de validation, d'observabilité et de surveillance des performances tels que EFA Prometheus Exporter et Nvidia Nsight Systems.

AWSome Inférence

Ce référentiel fournit des architectures de référence et des cas de test pour optimiser les solutions d'inférence AWS, en mettant l'accent sur Amazon EKS et les EC2 instances accélérées. Il inclut des configurations d'infrastructure pour les clusters VPC et EKS, des projets pour des frameworks tels que NIMs NVIDIA, TensorRT-LLM, Triton Inference Server et, avec des exemples pour des modèles tels que Llama3-8B RayService et Llama 3.1 405B. Propose des déploiements multi-nœuds utilisant K8s LeaderWorkerSet, l'autoscaling d'EKS, le multi-instance GPUs (MIG) et des cas d'utilisation réels tels qu'un bot audio pour l'ASR, l'inférence et le TTS.

Didacticiels

Si vous souhaitez configurer des plateformes et des frameworks de Machine Learning dans EKS, consultez les didacticiels décrits dans cette section. Ces didacticiels couvrent tout, des modèles permettant d'utiliser au mieux les processeurs GPU au choix des outils de modélisation en passant par la création de frameworks pour des industries spécialisées.

Créez des plateformes d'IA génératives sur EKS

- Déployez des modèles d'IA générative sur Amazon EKS
- Création de JupyterHub plateformes multi-locataires sur Amazon EKS

Exécutez des frameworks d'IA générative spécialisés sur EKS

- Accélérez vos charges de travail de formation distribuées basées sur l'IA générative avec le NeMo framework NVIDIA sur Amazon EKS
- Exécution TorchServe sur Amazon Elastic Kubernetes Service

Optimisez les performances du GPU NVIDIA pour le ML sur EKS

- Mettez en œuvre le partage de GPU pour utiliser efficacement NVIDIA GPUs pour vos clusters EKS :

  Partage de GPU sur Amazon EKS avec le découpage temporel et les instances accélérées NVIDIA EC2
- Utilisez les microservices Multi-Instance GPUs (MIGs) et NIM pour exécuter plus de pods par GPU sur vos clusters EKS :

Optimisation de l'utilisation du GPU avec le GPU multi-instance (MIG) de NVIDIA sur Amazon EKS : utilisation d'un plus grand nombre de modules par GPU pour des performances améliorées

- Créez et déployez un système d'apprentissage automatique évolutif sur Kubernetes avec Kubeflow sur AWS

Exécuter des charges de travail d'encodage vidéo sur EKS

- Diffusion de contenu vidéo fractionné GPUs dans des conteneurs sur Amazon EKS

Accélérez le chargement des images pour les charges de travail d'inférence

- Comment H2O.ai a optimisé et sécurisé son AI/ML infrastructure avec Karpenter et Bottlerocket

Surveillance des charges de travail ML

- Surveillance des charges de travail du GPU sur Amazon EKS à l'aide de services open source AWS gérés
- Activez les métriques GPU basées sur les pods dans Amazon CloudWatch

Gestion des versions sur Amazon EKS

Cette section est conçue pour vous aider à découvrir comment le versionnement de Kubernetes fonctionne dans Amazon Elastic Kubernetes Service (EKS). Vous trouverez des informations sur les versions prises en charge, nos politiques de dépréciation et de support, les processus de mise à niveau et les principales considérations relatives à la gestion des versions de votre cluster.

Rubriques

- Comprendre le cycle de vie des versions de Kubernetes sur EKS
- Afficher les versions de la plateforme Amazon EKS pour chaque version de Kubernetes

Comprendre le cycle de vie des versions de Kubernetes sur EKS

Kubernetes évolue rapidement avec de nouvelles fonctionnalités, des mises à jour de conception et des corrections de bugs. La communauté publie de nouvelles versions mineures de Kubernetes (telles que`1.33`) en moyenne une fois tous les quatre mois. Amazon EKS suit le cycle de publication en amont et d'obsolescence pour les versions mineures. À mesure que de nouvelles versions Kubernetes deviennent disponibles dans Amazon EKS, nous vous recommandons de mettre à jour de manière proactive vos clusters pour utiliser la dernière version disponible.

Une version mineure bénéficie d'un support standard dans Amazon EKS pendant les 14 premiers mois suivant sa sortie. Une fois qu'une version a dépassé la date de fin de support standard, elle passe en support étendu pour les 12 prochains mois. Le support étendu vous permet de rester plus longtemps sur une version spécifique de Kubernetes moyennant un coût supplémentaire par heure de cluster. Si vous n'avez pas mis à jour votre cluster avant la fin de la période de support étendu, celui-ci est automatiquement mis à niveau vers la plus ancienne version étendue actuellement prise en charge.

Le support étendu est activé par défaut. Pour le désactiver, voir Désactiver le support étendu d'EKS.

Nous vous recommandons de créer votre cluster avec la dernière version de Kubernetes disponible prise en charge par Amazon EKS. Si votre application nécessite une version spécifique de Kubernetes, vous pouvez sélectionner des versions plus anciennes. Vous pouvez créer de nouveaux clusters Amazon EKS sur n'importe quelle version proposée dans le cadre du support standard ou étendu.

Versions disponibles bénéficiant du support standard

Les versions de Kubernetes suivantes sont actuellement disponibles dans le support standard d'Amazon EKS :

- `1.33`
- `1.32`
- `1.31`
- `1.30`

Pour connaître les modifications importantes à prendre en compte pour chaque version du support standard, consultez le support standard des versions de Kubernetes.

Versions disponibles bénéficiant du support étendu

Les versions de Kubernetes suivantes sont actuellement disponibles dans le cadre du support étendu d'Amazon EKS :

- `1.29`
- `1.28`
- `1.27`

Pour connaître les modifications importantes à prendre en compte pour chaque version du support étendu, voir Support étendu des versions de Kubernetes.

Calendrier de publication Amazon EKS Kubernetes

Le tableau suivant indique les dates de sortie et de support importantes à prendre en compte pour chaque version de Kubernetes. La facturation du support étendu commence au début de la journée où la version atteint la fin du support standard, dans le fuseau horaire UTC+0. Les dates indiquées dans le tableau suivant utilisent le fuseau horaire UTC+0.

> **Note**
>
> Les dates avec seulement un mois et une année sont approximatives et sont mises à jour avec une date exacte quand elles sont connues.

Pour recevoir des notifications concernant toutes les modifications apportées au fichier source sur cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante avec un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/
kubernetes-versions.adoc.atom
```

Version de Kubernetes	Version en amont	Version d'Amazon EKS	Date de fin du support standard	Date de fin du support étendu
1.33	23 avril 2025	29 mai 2025	29 juillet 2026	29 juillet 2027
1.32	11 décembre 2024	23 janvier 2025	23 mars 2026	23 mars 2027
1.31	13 août 2024	26 septembre 2024	26 novembre 2025	26 novembre 2026
1.30	17 avril 2024	23 mai 2024	23 juillet 2025	23 juillet 2026
1.29	13 décembre 2023	23 janvier 2024	23 mars 2025	23 mars 2026
1.28	15 août 2023	26 septembre 2023	26 novembre 2024	26 novembre 2025
1.27	11 avril 2023	24 mai 2023	24 juillet 2024	24 juillet 2025

Obtenir des informations de version avec la AWS CLI

Vous pouvez utiliser la AWS CLI pour obtenir des informations sur les versions de Kubernetes disponibles sur EKS, telles que la date de fin du Support standard.

Pour récupérer des informations sur les versions de Kubernetes disponibles sur EKS à l'aide de la CLI AWS

1. Ouvrez votre terminal.
2. Assurez-vous que la AWS CLI est installée et configurée. Pour plus d'informations, voir Installation ou mise à jour vers la dernière version de la CLI.

3. Exécutez la commande suivante :

```
aws eks describe-cluster-versions
```

4. La commande renverra une sortie JSON avec des détails sur les versions de cluster disponibles. Voici un exemple de sortie :

```
{
    "clusterVersions": [
        {
            "clusterVersion": "1.31",
            "clusterType": "eks",
            "defaultPlatformVersion": "eks.21",
            "defaultVersion": true,
            "releaseDate": "2024-09-25T17:00:00-07:00",
            "endOfStandardSupportDate": "2025-11-25T16:00:00-08:00",
            "endOfExtendedSupportDate": "2026-11-25T16:00:00-08:00",
            "status": "STANDARD_SUPPORT",
            "kubernetesPatchVersion": "1.31.3"
        }
    ]
}
```

La sortie fournit les informations suivantes pour chaque version de cluster :

- `clusterVersion`: La version Kubernetes du cluster EKS
- `clusterType`: le type de cluster (par exemple, « eks »)
- `defaultPlatformVersion`: version par défaut de la plateforme EKS
- `defaultVersion`: s'il s'agit de la version par défaut
- `releaseDate`: date de sortie de cette version
- `endOfStandardSupportDate`: date à laquelle le support standard prend fin
- `endOfExtendedSupportDate`: date à laquelle le support étendu prend fin
- `status`: l'état de support actuel de la version, tel que `STANDARD_SUPPORT` ou `EXTENDED_SUPPORT`
- `kubernetesPatchVersion`: La version spécifique du correctif Kubernetes

Version d'Amazon EKS FAQs

Combien de versions de Kubernetes sont disponibles dans le cadre du support standard ?

Conformément au soutien de la communauté Kubernetes pour les versions de Kubernetes, Amazon EKS s'engage à proposer un support pour au moins trois versions de Kubernetes à tout moment. Nous annoncerons la date de fin du support standard d'une version mineure de Kubernetes donnée au moins 60 jours à l'avance. En raison du processus de qualification et de publication d'Amazon EKS pour les nouvelles versions de Kubernetes, la date de fin du support standard d'une version de Kubernetes sur Amazon EKS sera postérieure à la date à laquelle le projet Kubernetes arrête de prendre en charge la version en amont.

Pendant combien de temps un Kubernetes bénéficie-t-il du support standard d'Amazon EKS ?

Une version de Kubernetes a bénéficié d'un support standard pendant 14 mois après avoir été disponible pour la première fois sur Amazon EKS. Cela est vrai même si Kubernetes en amont ne prend plus en charge une version disponible sur Amazon EKS. Nous rétroportons les correctifs de sécurité applicables aux versions de Kubernetes prises en charge sur Amazon EKS.

Suis-je averti lorsque le support standard prend fin pour une version de Kubernetes sur Amazon EKS ?

Oui. Si l'un des clusters de votre compte exécute la version dont le support approche de la fin, Amazon EKS envoie une notification via le AWS Health Dashboard environ 12 mois après la sortie de la version Kubernetes sur Amazon EKS. L'avis indique la date de fin de la prise en charge. Le délai est d'au moins 60 jours à compter de la date de l'avis.

Quelles fonctionnalités de Kubernetes sont prises en charge par Amazon EKS ?

Amazon EKS prend en charge toutes les fonctionnalités généralement disponibles (GA) de l'API Kubernetes. Les nouvelles versions bêta APIs ne sont pas activées dans les clusters par défaut. Cependant, la version bêta existante APIs et les nouvelles versions de la version bêta existante APIs continuent d'être activées par défaut. Les fonctionnalités alpha ne sont pas prises en charge.

Les groupes de nœuds gérés par Amazon EKS sont-ils automatiquement mis à jour en même temps que la version du plan de contrôle du cluster ?

Non Un groupe de nœuds géré crée des EC2 instances Amazon dans votre compte. Ces instances ne sont pas automatiquement mises à niveau lorsque vous ou Amazon EKS mettez à jour votre plan de contrôle. Pour de plus amples informations, veuillez consulter the section called “Mettre à jour”. Nous vous recommandons de conserver la même version Kubernetes sur votre plan de contrôle et vos nœuds.

Les groupes de nœuds autogérés sont-ils automatiquement mis à jour en même temps que la version du plan de contrôle du cluster ?

Non Un groupe de nœuds autogéré inclut des EC2 instances Amazon dans votre compte. Ces instances ne sont pas automatiquement mises à niveau lorsque vous ou Amazon EKS mettez à jour la version du plan de contrôle en votre nom. Aucun groupe de nœuds autogéré n'indique dans la console qu'il doit être mis à jour. Vous pouvez afficher la version `kubelet` installée sur un nœud en sélectionnant le nœud dans la boîte de dialogue Nœuds dans la Présentation de votre cluster pour déterminer quels nœuds doivent être mis à jour. Vous devez mettre à jour les nœuds manuellement. Pour de plus amples informations, veuillez consulter the section called "Méthodes de mise à jour".

Le projet Kubernetes teste la compatibilité entre le plan de contrôle et les nœuds pour un maximum de trois versions mineures. Par exemple, les nœuds `1.30` continuent à fonctionner tant qu'ils sont orchestrés par un plan de contrôle `1.33`. Cependant, il n'est pas recommandé d'exécuter un cluster avec des nœuds situés en permanence à trois versions mineures du plan de contrôle. Pour plus d'informations, consultez Version Kubernetes et politique de prise en charge d'asymétrie de version dans la documentation Kubernetes. Nous vous recommandons de conserver la même version Kubernetes sur votre plan de contrôle et vos nœuds.

Les pods exécutés sur Fargate sont-ils automatiquement mis à niveau avec une mise à niveau automatique de la version du plan de contrôle du cluster ?

Non Nous recommandons vivement d'exécuter les Fargate Pods dans le cadre d'un contrôleur de réplication, tel qu'un déploiement de Kubernetes. Procédez ensuite à un redémarrage progressif de tous les Fargate Pods. La nouvelle version du Fargate Pod est déployée avec `kubelet` une version identique à la version mise à jour de votre plan de contrôle de cluster. Pour plus d'informations, consultez Déploiements dans la documentation Kubernetes.

> **Important**
>
> Si vous mettez à jour le plan de contrôle, vous devez également mettre à jour les nœuds Fargate vous-même. Pour mettre à jour les nœuds Fargate, supprimez le Fargate Pod représenté par le nœud et redéployez le Pod. Le nouveau Pod est déployé avec une `kubelet` version identique à celle de votre cluster.

Quelles sont les versions de Kubernetes prises en charge pour les nœuds hybrides ?

Amazon EKS Hybrid Nodes prend en charge les mêmes versions de Kubernetes que les clusters Amazon EKS avec d'autres types de calcul de nœuds, y compris la prise en charge des versions standard et étendue de Kubernetes. Les nœuds hybrides ne sont pas automatiquement mis à niveau lorsque vous mettez à niveau la version de votre plan de contrôle et vous êtes responsable de la mise à niveau de vos nœuds hybrides. Pour de plus amples informations, veuillez consulter the section called "Mettre à niveau les nœuds hybrides".

Support étendu d'Amazon EKS FAQs

La terminologie du support standard et du support étendu est nouvelle pour moi. Que signifient ces termes ?

Le support standard d'une version de Kubernetes dans Amazon EKS commence lorsqu'une version de Kubernetes est publiée sur Amazon EKS et prend fin 14 mois après la date de sortie. Le support étendu pour une version de Kubernetes débutera immédiatement après la fin du support standard et se terminera au bout de 12 mois. Par exemple, le support standard pour les versions `1.23` dans Amazon EKS a pris fin le 11 octobre 2023. Le support étendu pour la version `1.23` a débuté le 12 octobre 2023 et s'est terminé le 11 octobre 2024.

Que dois-je faire pour bénéficier d'un support étendu pour les clusters Amazon EKS ?

Vous devrez activer le support étendu (voir Support étendu EKS) pour votre cluster en modifiant la politique de mise à niveau du cluster en EXTENDED. Par défaut, pour tous les clusters nouveaux et existants, la politique de mise à niveau est définie sur EXTENDED, sauf indication contraire. Consultez la politique de mise à niveau du cluster pour connaître la politique de mise à niveau de votre cluster. Le support standard débutera lors de la publication d'une version de Kubernetes sur Amazon EKS et prendra fin 14 mois après la date de sortie. Le support étendu pour une version de Kubernetes débutera immédiatement après la fin du support standard et se terminera au bout de 12 mois.

Pour quelles versions de Kubernetes puis-je bénéficier d'un support étendu ?

Vous pouvez exécuter des clusters sur n'importe quelle version jusqu'à 12 mois après la fin du support standard pour cette version. Cela signifie que chaque version sera prise en charge pendant 26 mois dans Amazon EKS (14 mois de support standard plus 12 mois de support étendu).

Et si je ne souhaite pas utiliser le support étendu ?

Si vous ne souhaitez pas être automatiquement inscrit au support étendu, vous pouvez mettre à niveau votre cluster vers une version de Kubernetes incluse dans le support standard d'Amazon EKS. Pour désactiver le support étendu, voir Désactiver le support étendu EKS. Remarque : Si vous désactivez le support étendu, votre cluster sera automatiquement mis à niveau à la fin du support standard.

Que se passera-t-il à la fin des 12 mois de support prolongé ?

Les clusters exécutés sur une version de Kubernetes ayant terminé son cycle de vie de 26 mois (14 mois de support standard plus 12 mois de support étendu) seront automatiquement mis à niveau vers la version suivante. La mise à niveau automatique inclut uniquement le plan de contrôle Kubernetes. Si vous avez des nœuds EKS Auto Mode, ils peuvent être automatiquement mis à jour. Les nœuds autogérés et les groupes de nœuds gérés par EKS resteront dans la version précédente.

Passé la date de fin de la période de support étendu, vous ne pouvez plus créer de nouveaux clusters Amazon EKS avec la version non prise en charge. Les plans de contrôle existants sont automatiquement mis à jour par Amazon EKS vers la version prise en charge la plus récente, par le biais d'un processus de déploiement progressif après la date de fin de la prise en charge. Après la mise à jour automatique du plan de contrôle, veillez à mettre à jour manuellement les modules complémentaires du cluster et EC2 les nœuds Amazon. Pour de plus amples informations, veuillez consulter the section called “Mettre à jour la version de Kubernetes”.

Quand exactement mon plan de contrôle est-il automatiquement mis à jour après la fin de la date d'assistance prolongée ?

Amazon EKS ne peut pas fournir de délais spécifiques. Les mises à jour automatiques peuvent intervenir à tout moment après la date de fin de la période de support étendu. Vous ne recevrez aucune notification avant la mise à jour. Nous vous recommandons de mettre à jour, de manière proactive, votre plan de contrôle sans vous fier au processus de mise à jour automatique Amazon EKS. Pour de plus amples informations, veuillez consulter the section called “Mettre à jour la version de Kubernetes”.

Puis-je laisser mon plan de contrôle sur une version de Kubernetes indéfiniment ?

Non La sécurité du cloud AWS est la priorité absolue. Après un certain temps (généralement un an), la communauté Kubernetes cesse de publier des correctifs CVE (Common Vulnerability and Exposures) et décourage la soumission de CVE pour les versions non prises en charge. Cela signifie que les vulnérabilités spécifiques à une ancienne version de Kubernetes peuvent

même ne pas être signalées. Les clusters sont dès lors exposés sans préavis et aucune option de remédiation en cas de vulnérabilité n'est possible. Dans ces conditions, Amazon EKS n'autorise pas le maintien des plans de contrôle sur une version dont le support étendu a expiré.

Y a-t-il des frais supplémentaires pour bénéficier d'un support étendu ?

Oui, des frais supplémentaires sont facturés pour les clusters Amazon EKS exécutés dans le cadre du support étendu. Pour en savoir plus sur les tarifs, consultez le support étendu d'Amazon EKS pour la tarification des versions Kubernetes sur le AWS blog ou sur notre page de tarification.

Qu'est-ce qui est inclus dans le support étendu ?

Les clusters Amazon EKS bénéficiant du Support étendu reçoivent des correctifs de sécurité permanents pour le plan de contrôle Kubernetes. En outre, Amazon EKS publiera des correctifs pour Amazon VPC CNI `kube-proxy` et des modules complémentaires CoreDNS pour les versions Extended Support. Amazon EKS publiera également des correctifs pour les versions AWS publiées d'Amazon EKS optimisées AMIs pour Amazon Linux, Bottlerocket et Windows, ainsi que pour les nœuds Amazon EKS Fargate pour ces versions. Tous les clusters concernés par le Support étendu continueront d'avoir accès au support technique auprès de AWS.

Existe-t-il des limites aux correctifs pour les composants autres que Kubernetes dans le cadre du support étendu ?

Bien que le support étendu couvre tous les composants spécifiques de Kubernetes AWS, il ne fournira à tout moment qu'un support pour AWS Amazon EKS publié et optimisé pour AMIs Amazon Linux, Bottlerocket et Windows. Cela signifie que vous aurez potentiellement de nouveaux composants (système d'exploitation ou noyau, par exemple) sur votre AMI optimisée pour Amazon EKS lorsque vous utiliserez le support étendu. Par exemple, une fois qu'Amazon Linux 2 aura atteint la fin de son cycle de vie en 2025, Amazon Linux optimisé pour Amazon EKS AMIs sera développé à l'aide d'un système d'exploitation Amazon Linux plus récent. Amazon EKS annoncera et documentera les anomalies importantes du cycle de vie du support, telles que celle-ci, pour chaque version de Kubernetes.

Puis-je créer de nouveaux clusters à l'aide d'une version bénéficiant d'un support étendu ?

Oui.

Consultez les notes de publication des versions de Kubernetes sur le support standard

Cette rubrique présente les modifications importantes à prendre en compte pour chaque version de Kubernetes dans le support standard. Lors de la mise à niveau, examinez attentivement les modifications apportées entre l'ancienne et la nouvelle version de votre cluster.

Kubernetes 1.33

Kubernetes `1.33` est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes`1.33`, consultez l'annonce de sortie officielle.

> **Important**
>
> - L'API Kubernetes bêta d'allocation dynamique des ressources est activée.
> - Cette API bêta améliore l'expérience de planification et de surveillance des charges de travail nécessitant des ressources telles que GPUs.
> - L'API bêta est définie par la communauté Kubernetes et pourrait changer dans les futures versions de Kubernetes.
> - Lisez attentivement les étapes relatives aux fonctionnalités dans la documentation de Kubernetes pour comprendre les implications de l'utilisation de la version bêta. APIs
> - AWS ne publie pas d'AMI Amazon Linux 2 optimisée pour EKS pour Kubernetes 1.33.
> - AWS vous encourage à migrer vers Amazon Linux 2023. Découvrez comment the section called "Passez à la version AL2 023".
> - Pour de plus amples informations, veuillez consulter the section called "Obsolète de l'AMI Amazon Linux 2".

- Redimensionnement des ressources du module sur place (version bêta) : le redimensionnement des ressources sur place a été promu en version bêta, ce qui permet des mises à jour dynamiques des ressources du processeur et de la mémoire des pods existants sans redémarrage. Cela permet une mise à l'échelle verticale des charges de travail dynamiques sans interruption de service et un ajustement fluide des ressources en fonction des modèles de trafic.
- Les conteneurs annexes sont désormais stables : les conteneurs annexes sont devenus stables, utilisant des sidecars en tant que conteneurs d'initialisation spéciaux `restartPolicy: Always`

qui démarrent avant les conteneurs d'applications, fonctionnent tout au long du cycle de vie du pod et supportent les sondes pour la signalisation de l'état de fonctionnement.

- Pour plus d'informations, consultez la section Sidecar Containers dans la documentation de Kubernetes.

- Obsolète de l'API Endpoints : L'API Endpoints est désormais officiellement obsolète et renvoie des avertissements en cas d'accès. Migrez les charges de travail et les scripts pour utiliser plutôt l' EndpointSlices API, qui prend en charge des fonctionnalités modernes telles que le réseau à double pile et en gère plusieurs par service. EndpointSlices
  - Pour plus d'informations, consultez Kubernetes v1.33 : Continuing the transition from Endpoints to EndpointSlice the Kubernetes blog.
- Support de l'adaptateur Elastic Fabric : le groupe de sécurité par défaut pour les clusters Amazon EKS prend désormais en charge le trafic Elastic Fabric Adapter (EFA). Le groupe de sécurité par défaut possède une nouvelle règle sortante qui autorise le trafic EFA avec la destination du même groupe de sécurité. Cela autorise le trafic EFA au sein du cluster.
  - Pour plus d'informations, consultez l'adaptateur Elastic Fabric pour les charges de travail AI/ML et HPC sur Amazon EC2 dans le guide de l'utilisateur d'Amazon Elastic Compute Cloud.

Pour le journal complet des `1.33` modifications de Kubernetes, voir -1.33.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.32

Kubernetes `1.32` est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes`1.32`, consultez l'annonce de sortie officielle.

> **Important**
>
> - La version `flowcontrol.apiserver.k8s.io/v1beta3` API de FlowSchema et PriorityLevelConfiguration a été supprimée dans la version`1.32`. Si vous les utilisez APIs, vous devez mettre à jour vos configurations pour utiliser la dernière version prise en charge avant de procéder à la mise à niveau.
> - ServiceAccount `metadata.annotations[kubernetes.io/enforce-mountable-secrets]`a été obsolète dans sa version `1.32` et sera supprimée dans une future version mineure de Kubernetes. Il est recommandé d'utiliser des espaces de noms distincts pour isoler l'accès aux secrets montés.

- La version Kubernetes `1.32` est la dernière version pour laquelle Amazon EKS publiera Amazon Linux 2 (). AL2 AMIs `1.33`À partir de la version, Amazon EKS continuera à publier Amazon Linux 2023 (AL2023) et sur la base de Bottlerocket. AMIs

- La fonctionnalité Memory Manager est passée au statut généralement disponible (GA) dans la version Kubernetes. `1.32` Cette amélioration permet une allocation de mémoire plus efficace et plus prévisible pour les applications conteneurisées, ce qui est particulièrement avantageux pour les charges de travail ayant des exigences de mémoire spécifiques.
- PersistentVolumeClaims (PVCs) créé à StatefulSets ce jour inclut une fonctionnalité de nettoyage automatique. Lorsqu'ils ne PVCs sont plus nécessaires, ils sont automatiquement supprimés tout en maintenant la persistance des données pendant les StatefulSet mises à jour et les opérations de maintenance des nœuds. Cette fonctionnalité simplifie la gestion du stockage et permet d'éviter les orphelins PVCs dans votre cluster.
- La fonctionnalité de sélection de champs de ressources personnalisés a été introduite, permettant aux développeurs d'ajouter des sélecteurs de champs aux ressources personnalisées. Cette fonctionnalité fournit les mêmes fonctionnalités de filtrage disponibles pour les objets Kubernetes intégrés aux ressources personnalisées, permettant un filtrage des ressources plus précis et plus efficace et promouvant de meilleures pratiques de conception d'API.

Pour le journal des `1.32` modifications complet de Kubernetes, voir -1.32.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Modifications d'authentification anonymes

À partir d'Amazon EKS`1.32`, l'authentification anonyme est limitée aux points de terminaison de vérification de l'état du serveur d'API suivants :

- `/healthz`
- `/livez`
- `/readyz`

Les demandes adressées à tout autre point de terminaison utilisant l'`system:unauthenticated`utilisateur recevront une réponse `401 Unauthorized` HTTP. Cette amélioration de la sécurité permet d'empêcher l'accès involontaire au cluster qui pourrait se produire en raison de politiques RBAC mal configurées.

> **Note**
>
> Le rôle `public-info-viewer` RBAC continue de s'appliquer aux points de terminaison du bilan de santé répertoriés ci-dessus.

Obsolète de l'AMI Amazon Linux 2

Pour les versions 1.33 et ultérieures de Kubernetes, EKS ne fournira pas d'Amazon Linux 2 optimisées prédéfinies () Amazon Machine Images ()AL2. AMIs

AWS suggère d'adopter le mode automatique EKS ou de migrer vers un système d'exploitation plus récent, tel qu'Amazon Linux 2023 (AL2023) ou Bottlerocket.

- the section called "Migrer depuis MNGs"
- the section called "Passez à la version AL2 023"
- the section called "Bottlerocket"

> **Note**
>
> Cette mise à jour s'applique à EKS Optimized. AL2 AMIs Pour plus d'informations sur le système d'exploitation lui-même, consultez Amazon Linux 2 FAQs.

Kubernetes 1.31

Kubernetes `1.31` est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes`1.31`, consultez l'annonce de sortie officielle.

> **Important**
>
> - L'indicateur kubelet, `--keep-terminated-pod-volumes` obsolète depuis 2017, a été supprimé lors de la publication de la version. `1.31` Cette modification a un impact sur la façon dont les volumes de pods terminés sont gérés par le kubelet. Si vous utilisez cet indicateur dans les configurations de vos nœuds, vous devez mettre à jour vos scripts de démarrage et vos modèles de lancement pour le supprimer avant la mise à niveau.

- La porte de `VolumeAttributesClass` fonctionnalités bêta et la ressource API sont activées dans la version Amazon EKS`1.31`. Cette fonctionnalité permet aux opérateurs de clusters de modifier les propriétés mutables des volumes persistants (PVs) gérés par des pilotes CSI compatibles, notamment le pilote Amazon EBS CSI. Pour tirer parti de cette fonctionnalité, assurez-vous que votre pilote CSI prend en charge la `VolumeAttributesClass` fonctionnalité (pour le pilote Amazon EBS CSI, passez à la version `1.35.0` ou à une version ultérieure pour activer automatiquement la fonctionnalité). Vous pourrez créer `VolumeAttributesClass` des objets pour définir les attributs de volume souhaités, tels que le type de volume et le débit, et les associer à vos demandes de volume persistantes (PVCs). Consultez la documentation officielle de Kubernetes ainsi que la documentation de votre pilote CSI pour plus d'informations.
  - Pour plus d'informations sur le pilote Amazon EBS CSI, consultez. the section called "Amazon EBS"
- Le support de Kubernetes pour est AppArmordevenu stable et est désormais généralement disponible pour un usage public. Cette fonctionnalité vous permet de protéger vos conteneurs AppArmor en définissant le `appArmorProfile.type` champ dans le conteneur`securityContext`. Avant la version de Kubernetes`1.30`, il AppArmor était contrôlé par des annotations. À partir de la version`1.30`, elle est contrôlée à l'aide de champs. Pour tirer parti de cette fonctionnalité, nous vous recommandons de vous éloigner des annotations et d'utiliser le `appArmorProfile.type` champ pour garantir la compatibilité de vos charges de travail.
- La fonctionnalité de transition de PersistentVolume dernière phase est devenue stable et est désormais généralement disponible pour une utilisation publique dans la version Kubernetes. `1.31` Cette fonctionnalité introduit un nouveau champ, dans le `.status.lastTransitionTime` PersistentVolumeStatus, qui fournit un horodatage indiquant la date à laquelle une PersistentVolume dernière phase est passée à une phase différente. Cette amélioration permet un meilleur suivi et une meilleure gestion PersistentVolumes, en particulier dans les scénarios où il est important de comprendre le cycle de vie des volumes.

Pour le journal complet des `1.31` modifications de Kubernetes, voir -1.31.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.30

Kubernetes `1.30` est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes`1.30`, consultez l'annonce de sortie officielle.

> **Important**
>
> - À partir de la version Amazon EKS `1.30` ou plus récente, tous les groupes de nœuds gérés nouvellement créés utiliseront automatiquement par défaut Amazon Linux 2023 (AL2023) comme système d'exploitation des nœuds. Auparavant, les nouveaux groupes de nœuds étaient par défaut Amazon Linux 2 (AL2). Vous pouvez continuer à l'utiliser AL2 en le choisissant comme type d'AMI lors de la création d'un nouveau groupe de nœuds.
> - Pour plus d'informations sur la migration de la AL2 version AL2 023, consultez. the section called "Passez à la version AL2 023"
> - Pour plus d'informations sur Amazon Linux, consultez Comparing AL2 et AL2 023 dans le guide de l'utilisateur Amazon Linux.
> - Pour plus d'informations sur la spécification du système d'exploitation d'un groupe de nœuds gérés, consultez. the section called "Création"

- Avec Amazon EKS`1.30`, l'`topology.k8s.aws/zone-id`étiquette est ajoutée aux nœuds de travail. Vous pouvez utiliser la zone de disponibilité IDs (AZ IDs) pour déterminer l'emplacement des ressources d'un compte par rapport aux ressources d'un autre compte. Pour plus d'informations, consultez la section Zone de disponibilité IDs de vos AWS ressources dans le Guide de l'utilisateur de la AWS RAM.
- À partir de `1.30` là, Amazon EKS n'inclut plus l'`default`annotation sur la `gp2 StorageClass` ressource appliquée aux clusters nouvellement créés. Cela n'a aucun impact si vous référencez cette classe de stockage par son nom. Vous devez prendre des mesures si vous comptez sur une valeur par défaut `StorageClass` dans le cluster. Vous devez les référencer `StorageClass` par leur nom`gp2`. Vous pouvez également déployer la classe de stockage par défaut recommandée par Amazon EBS en définissant le `defaultStorageClass.enabled` paramètre sur true lors de l'installation de la version `1.31.0` ou ultérieure du`aws-ebs-csi-driver add-on`.
- La politique IAM minimale requise pour le rôle IAM du cluster Amazon EKS a changé. L'action `ec2:DescribeAvailabilityZones` est requise. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM du cluster".

Pour le journal complet des `1.30` modifications de Kubernetes, consultez -1.30.md. https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Consultez les notes de mise à jour des versions de Kubernetes sur le support étendu

Amazon EKS prend en charge les versions de Kubernetes plus longtemps qu'elles ne sont prises en charge en amont, avec un support standard pour les versions mineures de Kubernetes pendant 14 mois à compter de leur publication dans Amazon EKS, et un support étendu pour les versions mineures de Kubernetes pour 12 mois de support supplémentaires (26 mois au total par version).

Cette rubrique présente les modifications importantes à prendre en compte pour chaque version de Kubernetes bénéficiant d'un support étendu. Lors de la mise à niveau, examinez attentivement les modifications apportées entre l'ancienne et la nouvelle version de votre cluster.

Kubernetes 1.29

Kubernetes `1.29` est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes`1.29`, consultez l'annonce de sortie officielle.

> **Important**
>
> - Les versions d'`flowcontrol.apiserver.k8s.io/v1beta2`API obsolètes de `FlowSchema` et ne `PriorityLevelConfiguration` sont plus servies dans la version Kubernetes. `1.29` Si vous avez des manifestes ou un logiciel client qui utilise le groupe d'API bêta obsolète, vous devez les modifier avant de passer à la version. `1.29`

- Le `.status.kubeProxyVersion` champ pour les objets de nœud est désormais obsolète, et le projet Kubernetes propose de supprimer ce champ dans une future version. Le champ obsolète n'est pas précis et a toujours été géré par `kubelet` - qui ne connaît pas réellement la `kube-proxy` version, ni même si elle `kube-proxy` est en cours d'exécution. Si vous avez utilisé ce champ dans un logiciel client, arrêtez. Les informations ne sont pas fiables et le champ est désormais obsolète.
- Dans Kubernetes, `1.29` afin de réduire la surface d'attaque potentielle, la `LegacyServiceAccountTokenCleanUp` fonctionnalité identifie les anciens jetons secrets générés automatiquement comme non valides s'ils n'ont pas été utilisés depuis longtemps (1 an par défaut) et les supprime automatiquement si aucune tentative d'utilisation n'est effectuée pendant une longue période après avoir été marqués comme non valides (1 an supplémentaire par défaut). Pour identifier ces jetons, vous pouvez exécuter :

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

Pour le journal des 1.29 modifications complet de Kubernetes, consultez -1.29.md# 1280. https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG changelog-since-v

Kubernetes 1.28

Kubernetes 1.28 est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes1.28, consultez l'annonce de sortie officielle.

- Kubernetes v1.28 a étendu l'écart pris en charge entre les composants du nœud principal et du plan de contrôle d'une version mineure, de n-2 àn-3, afin que les composants du nœud (kubeletetkube-proxy) de la version secondaire prise en charge la plus ancienne puissent fonctionner avec les composants du plan de contrôle (kube-apiserver,, kube-schedulerkube-controller-manager,cloud-controller-manager) de la version mineure prise en charge la plus récente.
- Les métriques force_delete_pods_total et force_delete_pod_errors_total du Pod GC Controller ont été améliorées pour tenir compte de toutes les suppressions de pods forcées. Une raison est ajoutée à la métrique pour indiquer si le pod est supprimé de force parce qu'il est fermé, orphelin, altéré ou parce qu'il s'arrête et n'est out-of-service pas planifié.
- Le contrôleur PersistentVolume (PV) a été modifié pour attribuer automatiquement une StorageClass par défaut à toute PersistentVolumeClaim non associée dont le paramètre storageClassName n'est pas défini. En outre, le mécanisme de validation des PersistentVolumeClaim admissions au sein du serveur API a été ajusté pour permettre de changer les valeurs d'un état non défini à un StorageClass nom réel.

Pour le journal des 1.28 modifications complet de Kubernetes, consultez -1.28.md# 1270. https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG changelog-since-v

Kubernetes 1.27

Kubernetes 1.27 est désormais disponible sur Amazon EKS. Pour plus d'informations sur Kubernetes1.27, consultez l'annonce de sortie officielle.

Important

- La prise en charge des annotations alpha `seccomp.security.alpha.kubernetes.io/pod` et `container.seccomp.security.alpha.kubernetes.io`des annotations `seccomp` a été supprimée. Les annotations alpha `seccomp` sont devenues obsolètes en `1.19`, avec leur suppression de `1.27`, les champs `seccomp` ne seront plus remplis automatiquement pour les `Pods` avec les annotations `seccomp`. Utilisez plutôt le champ `securityContext.seccompProfile` réservé aux conteneurs ou `Pods` pour configurer les profils `seccomp`. Pour vous assurer que vous utilisez la base d'annotations alpha `seccomp` obsolètes dans votre cluster, exécutez la commande suivante :

```
kubectl get pods --all-namespaces -o json | grep
 -E 'seccomp.security.alpha.kubernetes.io/pod|
container.seccomp.security.alpha.kubernetes.io'
```

- L'argument de ligne de commande `--container-runtime` pour `kubelet` a été supprimé. L'environnement d'exécution du conteneur par défaut pour Amazon EKS a été `containerd` défini depuis`1.24`, ce qui élimine le besoin de spécifier le temps d'exécution du conteneur. À partir de `1.27`, Amazon EKS ignorera l'argument `--container-runtime` transmis à tous les scripts d'amorçage. Il est important de ne pas transmettre cet argument `--kubelet-extra-args` à afin d'éviter des erreurs lors du processus de démarrage du nœud. Vous devez supprimer l'argument `--container-runtime` de tous vos flux de travail de création de nœuds et de vos scripts de génération.

- `kubelet`Dans Kubernetes, la valeur par défaut `1.27` a été augmentée à et `kubeAPIQPS` à`50`. `kubeAPIBurst` `100` Ces améliorations permettent à `kubelet` de gérer un plus grand volume de requêtes d'API, améliorant ainsi les temps de réponse et les performances. Lorsque les demandes de `Pods` augmentent, en raison des exigences de mise à l'échelle, les valeurs par défaut révisées garantissent la capacité de `kubelet` à gérer efficacement l'augmentation de la charge de travail. Par conséquent, les lancements Pod sont plus rapides et les opérations de cluster sont plus efficaces.
- Vous pouvez utiliser une topologie Pod plus fine pour diffuser des politiques telles que `minDomain`. Ce paramètre vous permet de spécifier le nombre minimum de domaines sur lesquels vos `Pods` devez être répartis. `nodeAffinityPolicy` et `nodeTaintPolicy` permettent un niveau de granularité supplémentaire dans la gestion de la distribution Pod.

Ceci est conforme aux affinités des nœuds, aux rejets et au champ `matchLabelKeys` dans le `topologySpreadConstraints` de votre spécification `Pod's`. Cela permet de sélectionner des `Pods` pour étaler les calculs après une mise à niveau progressive.

- Kubernetes `1.27` a promu en version bêta un nouveau mécanisme de politique `StatefulSets` qui contrôle la durée de vie de leur `PersistentVolumeClaims` (). `PVCs` La nouvelle politique de rétention `PVC` vous permet de spécifier si les données `PVCs` générées à partir du modèle de spécification `StatefulSet` seront automatiquement supprimées ou retenues lors de la suppression du `StatefulSet` ou de la réduction des répliques dans le `StatefulSet`.
- L'option goaway-chance du serveur d'API Kubernetes permet d'éviter que les connexions `HTTP/2` client ne soient bloquées sur une seule instance de serveur d'API, en fermant une connexion de manière aléatoire. Lorsque la connexion est fermée, le client essaiera de se reconnecter et atterrira probablement sur un autre serveur d'API en raison de l'équilibrage de charge. La version `1.27` de Amazon EKS a activé l'indicateur `goaway-chance`. Si votre charge de travail exécutée sur le cluster Amazon EKS utilise un client qui n'est pas compatible avec HTTP GOAWAY, nous vous recommandons de mettre à jour la gestion de votre client en vous `GOAWAY` reconnectant à la fin de la connexion.

Pour le journal des `1.27` modifications complet de Kubernetes, consultez -1.27.md# 1260. https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG changelog-since-v

Afficher la période de support du cluster en cours

La section sur la période de support du cluster de la AWS console indique si votre cluster bénéficie actuellement d'un support standard ou étendu. Si la période de support de votre cluster est un support étendu, le support étendu d'EKS vous est facturé.

Pour plus d'informations sur le support standard et étendu, consultez eks/latest/userguide/ kubernetes -versions.html [Comprendre le cycle de vie des versions de Kubernetes sur EKS, type="documentation "].

1. Accédez à la page Clusters dans la section EKS de la AWS console. Vérifiez que la console est configurée dans la même AWS région que le cluster que vous souhaitez examiner.
2. Consultez la colonne « Période de support ». Si la valeur est Support standard jusqu'au..., l'assistance prolongée ne vous est pas facturée actuellement. Vous avez atteint la période de support standard. Si la valeur est Support étendu... ce cluster est actuellement facturé pour le support étendu.

> **Note**
>
> La période de Support ne peut pas être récupérée avec l' AWS API ou la CLI.

Afficher la politique de mise à niveau du cluster actuelle

La politique de mise à niveau du cluster détermine ce qui arrive à votre cluster lorsqu'il quitte la période de support standard. Si votre politique de mise à niveau l'est`EXTENDED`, le cluster ne sera pas automatiquement mis à niveau et bénéficiera d'un support étendu. Si votre politique de mise à niveau l'est`STANDARD`, elle sera automatiquement mise à niveau.

La politique de version d'Amazon EKS Controls for Kubernetes vous permet de choisir la fin du comportement de support standard pour vos clusters EKS. Grâce à ces contrôles, vous pouvez décider quels clusters doivent bénéficier du support étendu et quels clusters doivent être automatiquement mis à niveau à la fin du support standard pour une version de Kubernetes.

Une version mineure bénéficie d'un support standard dans Amazon EKS pendant les 14 premiers mois suivant sa sortie. Une fois qu'une version a dépassé la date de fin de support standard, elle passe en support étendu pour les 12 prochains mois. Le support étendu vous permet de rester plus longtemps sur une version spécifique de Kubernetes moyennant un coût supplémentaire par heure de cluster. Vous pouvez activer ou désactiver le support étendu pour un cluster EKS. Si vous désactivez le support étendu, votre cluster AWS sera automatiquement mis à niveau vers la version suivante à la fin du support standard. Si vous activez le support étendu, vous pouvez conserver la version actuelle moyennant des frais supplémentaires pendant une période limitée. Prévoyez de mettre régulièrement à niveau votre cluster Kubernetes, même si vous utilisez un support étendu.

Vous pouvez définir la politique de version pour les clusters nouveaux et existants à l'aide de la `supportType` propriété. Deux options peuvent être utilisées pour définir la politique de support des versions :

- **`STANDARD`** — Votre cluster EKS est éligible à la mise à niveau automatique à la fin du support standard. Ce paramètre ne vous facturera pas de frais de support prolongés, mais votre cluster EKS sera automatiquement mis à niveau vers la prochaine version de Kubernetes prise en charge dans le cadre du support standard.
- **`EXTENDED`** — Votre cluster EKS bénéficiera d'un support étendu une fois que la version Kubernetes aura atteint la fin du support standard. Vous devrez payer des frais d'assistance prolongés avec ce paramètre. Vous pouvez mettre à niveau votre cluster vers une version standard

de Kubernetes prise en charge pour ne plus avoir à payer de frais de support prolongés. Les clusters fonctionnant avec un support étendu pourront bénéficier d'une mise à niveau automatique à la fin du support étendu.

Le support étendu est activé par défaut pour les nouveaux clusters et les clusters existants. Vous pouvez voir si le support étendu est activé pour un cluster dans ou à l' AWS Management Console aide de l' AWS interface de ligne de commande.

> **Important**
>
> Si vous souhaitez que votre cluster conserve sa version actuelle de Kubernetes afin de bénéficier de la période de support étendue, vous devez activer la politique de mise à niveau du support étendu avant la fin de la période de support standard.

Vous ne pouvez définir la politique de support de version pour vos clusters que lorsqu'ils sont exécutés sur la version Kubernetes dans le cadre du support standard. Une fois que la version est entrée dans le support étendu, vous ne pourrez pas modifier ce paramètre tant que vous n'utiliserez pas une version en support standard.

Par exemple, si vous avez défini votre politique de prise en charge des versions`standard`, vous ne pourrez pas modifier ce paramètre une fois que la version de Kubernetes exécutée sur votre cluster aura atteint la fin du support standard. Si vous avez défini votre politique de prise en charge des versions`extended`, vous ne pourrez pas modifier ce paramètre une fois que la version de Kubernetes exécutée sur votre cluster aura atteint la fin du support standard. Pour modifier le paramètre de politique de prise en charge des versions, votre cluster doit être exécuté sur une version standard de Kubernetes prise en charge.

Afficher la politique de mise à niveau du cluster (AWS console)

1. Accédez à la page Clusters dans la section EKS de la AWS console. Vérifiez que la console est configurée dans la même AWS région que le cluster que vous souhaitez examiner.
2. Consultez la colonne Politique de mise à niveau. Si la valeur est Support standard, votre cluster ne bénéficiera pas du support étendu. Si la valeur est Support étendu, votre cluster entrera dans le champ Support étendu.

Afficher la politique de mise à niveau du cluster (AWS CLI)

1. Vérifiez que la AWS CLI est installée et que vous êtes connecté. Découvrez comment mettre à jour et installer la AWS CLI.
2. Déterminez le nom de votre cluster EKS. Définissez la CLI sur la même AWS région que votre cluster EKS.
3. Exécutez la commande suivante :

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. Si la valeur est`STANDARD`, votre cluster ne bénéficiera pas du support étendu. Si la valeur est`EXTENDED`, votre cluster bénéficiera d'un support étendu.

Ajoutez de la flexibilité pour planifier les mises à niveau des versions de Kubernetes en activant le support étendu d'EKS

Cette rubrique décrit comment définir la politique de mise à niveau d'un cluster EKS pour permettre un support étendu. La politique de mise à niveau d'un cluster EKS détermine ce qui se passe lorsqu'un cluster atteint la fin de la période de support standard. Si une politique de mise à niveau du cluster a activé le support étendu, elle entrera dans la période de support étendu à la fin de la période de support standard. Le cluster ne sera pas automatiquement mis à niveau à la fin de la période de support standard.

Les clusters pendant la période de support prolongée entraînent en fait des coûts plus élevés. Si un cluster dispose simplement d'une politique de mise à niveau définie pour permettre un support étendu, et qu'il se trouve par ailleurs dans la période de support standard, il encourt des coûts standard.

Si vous créez un cluster dans la AWS console, la politique de mise à niveau sera définie pour désactiver le support étendu. Si vous créez un cluster d'une autre manière, la politique de mise à niveau sera définie pour permettre un support étendu. Par exemple, le support étendu est activé pour les clusters créés avec l' AWS API.

Pour plus d'informations sur les politiques de mise à niveau, consultez la section Politique de mise à niveau du cluster.

> **Important**
>
> Si vous souhaitez que votre cluster conserve sa version actuelle de Kubernetes afin de bénéficier de la période de support étendue, vous devez activer la politique de mise à niveau du support étendu avant la fin de la période de support standard.
> Si vous n'activez pas le support étendu, votre cluster sera automatiquement mis à niveau.

Activer le support étendu EKS (AWS console)

1. Accédez à votre cluster EKS dans la AWS console. Sélectionnez l'onglet Vue d'ensemble sur la page d'informations du cluster.
2. Dans la section des paramètres de version de Kubernetes, sélectionnez Gérer.
3. Sélectionnez Support étendu, puis Enregistrer les modifications.

Activer le support étendu (AWS CLI) d'EKS

1. Vérifiez que la AWS CLI est installée et que vous êtes connecté. Découvrez comment mettre à jour et installer la AWS CLI.
2. Déterminez le nom de votre cluster EKS.
3. Exécutez la commande suivante :

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

Empêchez l'augmentation des coûts des clusters en désactivant le support étendu d'EKS

Cette rubrique décrit comment définir la politique de mise à niveau d'un cluster EKS afin de désactiver le support étendu. La politique de mise à niveau d'un cluster EKS détermine ce qui se passe lorsqu'un cluster atteint la fin de la période de support standard. Si une politique de mise à niveau de cluster a désactivé le support étendu, elle sera automatiquement mise à niveau vers la prochaine version de Kubernetes.

Pour plus d'informations sur les politiques de mise à niveau, consultez la section Politique de mise à niveau du cluster.

> **Important**
>
> Vous ne pouvez pas désactiver le support étendu une fois que votre cluster y a accédé. Vous ne pouvez désactiver le support étendu pour les clusters que sur le support standard. AWS recommande de mettre à niveau votre cluster vers une version pendant la période de support standard.

Désactiver le support étendu EKS (AWS console)

1. Accédez à votre cluster EKS dans la AWS console. Sélectionnez l'onglet Vue d'ensemble sur la page d'informations du cluster.
2. Dans la section des paramètres de version de Kubernetes, sélectionnez Gérer.
3. Sélectionnez Support standard, puis Enregistrer les modifications.

Désactiver le support étendu EKS (AWS CLI)

1. Vérifiez que la AWS CLI est installée et que vous êtes connecté. Découvrez comment mettre à jour et installer la AWS CLI.
2. Déterminez le nom de votre cluster EKS.
3. Exécutez la commande suivante :

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```

Afficher les versions de la plateforme Amazon EKS pour chaque version de Kubernetes

Les versions de la plateforme Amazon EKS représentent les fonctionnalités du plan de contrôle du cluster Amazon EKS, telles que les indicateurs de serveur d'API Kubernetes qui sont activés, ainsi que la version de correctif Kubernetes actuelle. Une ou plusieurs versions de plateforme Amazon

EKS sont associées à chaque version mineure de Kubernetes. Les versions de plateforme pour les différentes versions mineures de Kubernetes sont indépendantes. Vous pouvez récupérer la version actuelle de la plate-forme de votre cluster à l'aide de la AWS CLI ou AWS Management Console. Si vous avez un cluster local sur AWS Outposts, consultez the section called "Versions de la plateforme EKS" plutôt cette rubrique.

Lorsqu'une nouvelle version mineure de Kubernetes est disponible dans Amazon EKS, telle que la version 1.33, la version initiale de la plateforme Amazon EKS pour cette version mineure de Kubernetes commence à. `eks.1` Cependant, Amazon EKS publie régulièrement de nouvelles versions de plateforme pour activer de nouveaux paramètres de plan de contrôle Kubernetes et fournir des correctifs de sécurité.

Lorsque de nouvelles versions de plateforme Amazon EKS deviennent disponibles pour une version mineure :

- Le numéro de version de plateforme Amazon EKS est incrémenté (`eks.<n+1>`).
- Amazon EKS met automatiquement à niveau tous les clusters existants vers la dernière version de plateforme Amazon EKS pour les versions mineures Kubernetes correspondantes. Les mises à niveau automatiques de versions de plateforme Amazon EKS existantes sont déployées de façon incrémentielle. Le processus de déploiement peut prendre du temps. Si vous avez besoin immédiatement des fonctions de la dernière version de plateforme Amazon EKS, vous devez créer un nouveau cluster Amazon EKS.

  Si votre cluster a plus de deux versions de plate-forme en retard par rapport à la version actuelle, il est possible qu'Amazon EKS n'ait pas été en mesure de mettre à jour automatiquement votre cluster. Pour plus de détails sur la cause de ce problème, consultez the section called "La version de la plateforme Amazon EKS est inférieure de plus de deux versions à la version actuelle de la plateforme".
- Amazon EKS peut publier une nouvelle AMI de nœud avec une version de correctif correspondante. Cependant, toutes les versions de correctif sont compatibles entre le plan de contrôle et le nœud EKS AMIs pour une version mineure de Kubernetes donnée.

Les nouvelles versions de la plateforme Amazon EKS n'introduisent pas de modifications majeures ni n'entraînent d'interruptions de service.

Les clusters sont toujours créés avec la dernière version de plateforme Amazon EKS disponible (`eks.<n>`) pour la version Kubernetes spécifiée. Si vous mettez à jour votre cluster vers une

nouvelle version mineure de Kubernetes, votre cluster reçoit la version de plateforme Amazon EKS actuelle pour la version mineure Kubernetes vers laquelle vous avez effectué la mise à jour.

Les versions actuelles et récentes de plateforme Amazon EKS sont décrites dans les tableaux ci-dessous.

> **Note**
>
> AWS a récemment désactivé certaines versions de la plateforme publiées en juin 2024. Les versions de la plateforme présentaient des problèmes de stabilité. Aucune action n'est nécessaire.

Pour recevoir des notifications concernant toutes les modifications apportées au fichier source sur cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante avec un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/
platform-versions.adoc.atom
```

Version de Kubernetes `1.33`

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.33` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.33.2`	`eks.9`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.33 `eks.7` et 1.33 `eks.8` ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.33.1`	`eks.6`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
`1.33.1`	`eks.5`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
`1.33.1`	`eks.4`	Publication initiale de la version Kubernetes pour EKS. `1.33` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.33".	28 mai 2025

Version de Kubernetes **`1.32`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.32` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.32.6`	`eks.16`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.32 `eks.14`	30 juillet 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		et 1.32 `eks.15` ont été supprimées en interne et n'ont jamais été publiées.	
`1.32.5`	`eks.13`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
`1.32.5`	`eks.12`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
`1.32.5`	`eks.11`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
`1.32.3`	`eks.10`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
`1.32.3`	`eks.9`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.3	eks.8	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.32.3	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.32.3	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.32.2	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.32.2	eks.4	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.32.1	eks.3	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.32.0`	`eks.2`	Publication initiale de la version Kubernetes pour EKS. `1.32` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.32".	Janvier 2025

Version de Kubernetes **`1.31`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes de version `1.31` : `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`, `ObjectCount`.

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.31.10`	`eks.32`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.31 `eks.30` et 1.31 `eks.31` ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.9	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
1.31.9	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.31.9	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.31.7	eks.26	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.31.7	eks.25	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.31.7	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.7	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.31.7	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.31.6	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.31.6	eks.20	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.31.5	eks.19	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.31.5	eks.18	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.31.4`	`eks.17`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025
`1.31.2`	`eks.12`	Nouvelle version de la plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations apportées à l'observabilité du plan de contrôle. See the section called "Nœuds hybrides" and see Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024
`1.31.1`	`eks.6`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.31.0`	`eks.4`	Publication initiale de la version Kubernetes pour EKS. `1.31` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.31".	26 septembre 2024

Version de Kubernetes **`1.30`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.30` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.30.14`	`eks.40`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.30 `eks.38` et 1.30 `eks.39` ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
`1.30.13`	`eks.37`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.13	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.30.13	eks.35	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.30.11	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.30.11	eks.33	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.30.11	eks.32	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.30.11	eks.31	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.11	eks.30	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.30.10	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.30.10	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.30.9	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.30.8	eks.25	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025
1.30.8	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.7	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.30.6	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.30.6	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.30.6	eks.20	Nouvelle version de la plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations apportées à l'observabilité du plan de contrôle. See the section called "Nœuds hybrides" and see Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.5	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.30.4	eks.8	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024
1.30.3	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024
1.30.3	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.30.2	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.30.0`	`eks.2`	Publication initiale de la version Kubernetes pour EKS. `1.30` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.30".	23 mai 2024

Version de Kubernetes **`1.29`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.29` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.29.15`	`eks.43`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.29 `eks.41` et 1.29 `eks.42` ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
`1.29.15`	`eks.40`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.39	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.29.15	eks.38	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.29.15	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.29.15	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.29.15	eks.35	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.29.15	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.33	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.29.14	eks.32	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.29.14	eks.31	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.29.13	eks.30	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.29.13	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.29.12	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 janvier 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.29.12`	`eks.27`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025
`1.29.11`	`eks.26`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
`1.29.10`	`eks.25`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
`1.29.10`	`eks.24`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.10	eks.23	Nouvelle version de la plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations apportées à l'observabilité du plan de contrôle. See the section called "Nœuds hybrides" and see Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024
1.29.9	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.29.8	eks.13	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024
1.29.7	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.7	eks.11	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.29.6	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024
1.29.4	eks.7	Nouvelle version de la plateforme avec mise à l'échelle automatique de CoreDNS, correctifs de sécurité et améliorations. Pour plus d'informations sur la mise à l'échelle automatique de CoreDNS, consultez . the section called "Échelle adaptée à un trafic élevé"	16 mai 2024
1.29.3	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.29.1`	`eks.5`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 mars 2024
`1.29.1`	`eks.4`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 mars 2024
`1.29.1`	`eks.3`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 mars 2024
`1.29.0`	`eks.1`	Publication initiale de la version Kubernetes pour EKS. `1.29` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.29".	23 janvier 2024

Version de Kubernetes **`1.28`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.28` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.28.15	eks.49	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.28 eks.47 et 1.28 eks.48 ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
1.28.15	eks.46	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
1.28.15	eks.45	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.28.15	eks.44	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	Le 30 mai. 2025
1.28.15	eks.43	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.28.15	eks.42	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.28.15	eks.41	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.28.15	eks.40	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.28.15	eks.39	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.28.15	eks.38	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.28.15	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.28.15	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.28.15	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025
1.28.15	eks.33	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025
1.28.15	eks.32	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.28.15	eks.31	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.28.15	eks.30	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.28.15	eks.29	Nouvelle version de la plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations apportées à l'observabilité du plan de contrôle. See the section called "Nœuds hybrides" and see Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024
1.28.14	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.28.13	eks.19	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024
1.28.12	eks.18	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.28.11	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.28.11	eks.16	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024
1.28.9	eks.13	Nouvelle version de la plateforme avec mise à l'échelle automatique de CoreDNS, correctifs de sécurité et améliorations. Pour plus d'informations sur la mise à l'échelle automatique de CoreDNS, consultez . the section called "Échelle adaptée à un trafic élevé"	16 mai 2024
1.28.8	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.28.7	eks.11	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 mars 2024
1.28.7	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 mars 2024
1.28.6	eks.9	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 mars 2024
1.28.5	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2024
1.28.4	eks.6	Nouvelle version de la plateforme avec entrées d'accès, correctifs de sécurité et améliorations.	14 décembre 2023
1.28.4	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 décembre 2023

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.28.3`	`eks.4`	Nouvelle version de la plateforme avec Découvrez comment EKS Pod Identity permet aux pods d'accéder aux AWS services, aux correctifs de sécurité et aux améliorations.	10 novembre 2023
`1.28.3`	`eks.3`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 novembre 2023
`1.28.2`	`eks.2`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 octobre 2023
`1.28.1`	`eks.1`	Publication initiale de la version Kubernetes pour EKS. `1.28` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.28".	26 septembre 2023

Version de Kubernetes **1.27**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.27` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.27.16`	`eks.53`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Les versions 1.27 `eks.51` et 1.27 `eks.52` ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
`1.27.16`	`eks.50`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
`1.27.16`	`eks.49`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
`1.27.16`	`eks.48`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
`1.27.16`	`eks.47`	Nouvelle version de la plateforme	16 mai 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		avec des corrections de sécurité et des améliorations.	
1.27.16	eks.46	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.27.16	eks.45	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.27.16	eks.44	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.27.16	eks.43	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.27.16	eks.42	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.27.16	eks.41	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.27.16	eks.40	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.27.16	eks.38	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025
1.27.16	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025
1.27.16	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.27.16	eks.35	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.27.16	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.27.16	eks.33	Nouvelle version de la plateforme avec prise en charge des nœuds hybrides Amazon EKS, correctifs de sécurité et améliorations. Pour plus d'informations sur les nœuds hybrides Amazon EKS, consultezthe section called "Nœuds hybrides".	15 novembre 2024
1.27.16	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.27.16	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.27.16	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024
1.27.16	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.27.15	eks.20	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024
1.27.13	eks.17	Nouvelle version de la plateforme avec mise à l'échelle automatique de CoreDNS, correctifs de sécurité et améliorations. Pour plus d'informations sur la mise à l'échelle automatique de CoreDNS, consultez . the section called "Échelle adaptée à un trafic élevé"	16 mai 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.27.12	eks.16	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2024
1.27.11	eks.15	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 mars 2024
1.27.11	eks.14	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 mars 2024
1.27.10	eks.13	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 mars 2024
1.27.9	eks.11	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2024
1.27.8	eks.10	Nouvelle version de la plateforme avec entrées d'accès, correctifs de sécurité et améliorations.	14 décembre 2023

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.27.8	eks.9	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 décembre 2023
1.27.7	eks.8	Nouvelle version de la plateforme avec Découvrez comment EKS Pod Identity permet aux pods d'accéder aux AWS services, aux correctifs de sécurité et aux améliorations.	10 novembre 2023
1.27.7	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 novembre 2023
1.27.6	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 octobre 2023
1.27.4	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 août 2023

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.27.4`	`eks.4`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 juillet 2023
`1.27.3`	`eks.3`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 juin 2023
`1.27.2`	`eks.2`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 juin 2023
`1.27.1`	`eks.1`	Publication initiale de la version Kubernetes pour EKS. `1.27` Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.27".	24 mai 2023

Version de Kubernetes **`1.26`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.26` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.26.15	eks.51	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.26.15	eks.50	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.26.15	eks.49	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.26.15	eks.48	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.26.15	eks.47	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.26.15	eks.46	Nouvelle version de la plateforme avec des correctio	18 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		ns de sécurité et des améliorations.	
1.26.15	eks.45	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.26.15	eks.44	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.26.15	eks.43	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.26.15	eks.42	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.26.15	eks.40	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.26.15	eks.39	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025
1.26.15	eks.38	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.26.15	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.26.15	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.26.15	eks.35	Nouvelle version de la plateforme avec prise en charge des nœuds hybrides Amazon EKS, correctifs de sécurité et améliorations. Pour plus d'informations sur les nœuds hybrides Amazon EKS, consultezthe section called "Nœuds hybrides".	15 novembre 2024
1.26.15	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.26.15	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024
1.26.15	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.26.15	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.26.15	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024
1.26.15	eks.18	Nouvelle version de la plateforme avec mise à l'échelle automatique de CoreDNS, correctifs de sécurité et améliorations. Pour plus d'informations sur la mise à l'échelle automatique de CoreDNS, consultez . the section called "Échelle adaptée à un trafic élevé"	16 mai 2024
1.26.15	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.26.14	eks.16	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 mars 2024
1.26.14	eks.15	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 mars 2024
1.26.13	eks.14	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 mars 2024
1.26.12	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2024
1.26.11	eks.11	Nouvelle version de la plateforme avec entrées d'accès, correctifs de sécurité et améliorations.	14 décembre 2023
1.26.11	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 décembre 2023

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.26.10	eks.9	Nouvelle version de la plateforme avec Découvrez comment EKS Pod Identity permet aux pods d'accéder aux AWS services, aux correctifs de sécurité et aux améliorations.	10 novembre 2023
1.26.10	eks.8	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 novembre 2023
1.26.9	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 octobre 2023
1.26.7	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 août 2023
1.26.7	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 juillet 2023

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
`1.26.6`	`eks.4`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 juin 2023
`1.26.5`	`eks.3`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 juin 2023
`1.26.4`	`eks.2`	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	5 mai 2023
`1.26.2`	`eks.1`	Publication initiale de la version Kubernetes pour EKS. `1.26`	11 avril 2023

Obtenir la version actuelle de la plateforme

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation, choisissez Clusters.
3. Dans la liste des clusters, choisissez le nom du cluster pour vérifier la version de la plateforme.
4. Choisissez l'onglet Overview (Présentation).
5. La version de la plateforme est disponible dans la section Détails.
6. Déterminez le nom du cluster dont vous souhaitez vérifier la version de la plateforme.
7. Exécutez la commande suivante :

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

L'exemple qui suit illustre un résultat.

```
"eks.10"
```

Modifier la version de la plateforme

Vous ne pouvez pas modifier la version de plate-forme d'un cluster EKS. Lorsque de nouvelles versions de la plateforme Amazon EKS sont disponibles pour une version de Kubernetes, EKS met automatiquement à niveau tous les clusters existants vers la dernière version de la plateforme Amazon EKS pour la version Kubernetes correspondante. Les mises à niveau automatiques de versions de plateforme Amazon EKS existantes sont déployées de façon incrémentielle. Vous ne pouvez pas utiliser la AWS console ou la CLI pour modifier la version de la plateforme.

Si vous mettez à niveau votre version de Kubernetes, votre cluster passera à la version de plate-forme la plus récente pour la version de Kubernetes.

Résoudre les problèmes liés aux clusters et nœuds Amazon EKS

Ce chapitre traite de certaines erreurs courantes que vous pouvez rencontrer lorsque vous utilisez Amazon EKS, ainsi que des solutions. Si vous avez besoin de dépanner des zones spécifiques d'Amazon EKS, consultez les rubriques the section called "Résolution des problèmes", the section called "Résoudre les problèmes liés au connecteur EKS" et Dépannage d'ADOT à l'aide de modules complémentaires EKS.

Pour d'autres informations de dépannage, consultez le contenu du centre de connaissances sur Amazon Elastic Kubernetes Service sur Re:post. AWS

Capacité insuffisante

Si vous recevez le message d'erreur suivant lors de la tentative de création d'un cluster Amazon EKS, cela signifie que l'une des zones de disponibilité que vous avez spécifiées ne dispose pas d'une capacité suffisante pour prendre en charge un cluster.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted
Availability Zone, does not currently have sufficient capacity to support
the cluster. Retry and choose from these Availability Zones: region-1a,
region-1b, region-1c
```

Essayez à nouveau de créer votre cluster avec des sous-réseaux de votre VPC de cluster hébergés dans les zones de disponibilité renvoyés par ce message d'erreur.

Il existe des zones de disponibilité dans lesquelles un cluster ne peut pas résider. Comparez les zones de disponibilité dans lesquelles se trouvent vos sous-réseaux avec la liste des zones de disponibilité figurant dans la section Exigences et considérations relatives aux sous-réseaux.

Les nœuds ne parviennent pas à joindre le cluster

Quelques raisons courantes empêchent les nœuds de joindre le cluster :

- Si les nœuds sont des nœuds gérés, Amazon EKS ajoute des entrées à la `ConfigMap` `aws-auth` lorsque vous créez le groupe de nœuds. Si l'entrée a été supprimée ou modifiée, vous devez l'ajouter de nouveau. Pour plus d'informations, entrez `eksctl create iamidentitymapping`

`--help` dans votre terminal. Vous pouvez afficher vos `aws-auth` `ConfigMap` entrées actuelles *`my-cluster`* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :`eksctl get iamidentitymapping --cluster` *`my-cluster`* . L'ARN du rôle que vous spécifiez ne peut pas inclure de chemin autre que`/`. Par exemple, si le nom de votre rôle est`development/apps/my-role`, vous devrez le remplacer `my-role` lorsque vous spécifiez l'ARN du rôle. Assurez-vous que vous spécifiez l'ARN pour le rôle IAM du nœud (et non l'ARN du profil d'instance).

Si les nœuds sont autogérés et que vous n'avez pas créé d'entrées d'accès pour l'ARN du rôle IAM du nœud, exécutez les mêmes commandes répertoriées pour les nœuds gérés. Si vous avez créé une entrée d'accès pour l'ARN du rôle IAM de votre nœud, il se peut qu'elle ne soit pas correctement configurée dans l'entrée d'accès. Assurez-vous que l'ARN du rôle IAM du nœud (et non l'ARN du profil d'instance) est spécifié comme ARN principal dans votre entrée de `ConfigMap` `aws-auth` ou dans votre entrée d'accès. Pour plus d'informations sur les entrées d'accès, consultez the section called "Accorder des autorisations".

- Le AWS CloudFormation modèle ClusterNamede votre nœud ne correspond pas exactement au nom du cluster que vous souhaitez que vos nœuds rejoignent. La transmission d'une valeur incorrecte à ce champ entraîne une configuration incorrecte du `/var/lib/kubelet/kubeconfig` fichier du nœud et les nœuds ne rejoindront pas le cluster.
- Le nœud n'est pas étiqueté comme appartenant au cluster. La balise suivante doit être appliquée à vos nœuds, où *`my-cluster`* est remplacé par le nom de votre cluster.

Clé	Valeur
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

- Les nœuds peuvent ne pas être en mesure d'accéder au cluster à l'aide d'une adresse IP publique. Assurez-vous qu'une adresse IP publique est attribuée aux nœuds déployés dans les sous-réseaux publics. Dans le cas contraire, vous pouvez associer une adresse IP élastique à un nœud après son lancement. Pour plus d'informations, consultez Association d'une adresse IP élastique à une instance en cours d'exécution ou à une interface réseau. Si le sous-réseau public n'est pas défini pour attribuer automatiquement des adresses IP publiques aux instances qui y sont déployées, nous vous recommandons d'activer ce paramètre. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau. Si le nœud est déployé sur un sous-réseau privé, ce sous-réseau doit disposer d'une route vers une passerelle NAT à laquelle une adresse IP publique est attribuée.

- Le point de terminaison AWS STS de la AWS région dans laquelle vous déployez les nœuds n'est pas activé pour votre compte. Pour activer la région, voir Activation et désactivation du AWS STS dans une AWS région.
- Le nœud ne possède pas d'entrée DNS privée, ce qui entraîne une `node "" not found` erreur dans le `kubelet` journal. Assurez-vous que le VPC sur lequel le nœud est créé a des valeurs définies pour `domain-name` et `domain-name-servers` comme `Options` dans un `DHCP options set`. Les valeurs par défaut sont `domain-name:<region>.compute.internal` et `domain-name-servers:AmazonProvidedDNS`. Pour en savoir plus, consultez Jeux d'options DHCP dans le Guide de l'utilisateur Amazon VPC.
- Si les nœuds du groupe de nœuds gérés ne se connectent pas au cluster dans les 15 minutes, un problème de santé de type NodeCreationFailure « » sera émis et le statut de la console sera défini sur`Create failed`. Pour Windows AMIs dont les temps de lancement sont lents, ce problème peut être résolu à l'aide du lancement rapide.

Pour identifier et résoudre les problèmes courants qui empêchent les composants master de rejoindre un cluster, vous pouvez utiliser le runbook`AWSSupport-TroubleshootEKSWorkerNode`. Pour plus d'informations, consultez le manuel `AWSSupport-TroubleshootEKSWorkerNode` de référence du runbook AWS Systems Manager Automation.

Accès non autorisé ou refusé (`kubectl`)

Si vous recevez l'une des erreurs suivantes lors de l'exécution de `kubectl` commandes, cela signifie que vous n'avez pas correctement `kubectl` configuré Amazon EKS ou que les informations d'identification du principal IAM (rôle ou utilisateur) que vous utilisez ne correspondent pas à un nom d'utilisateur Kubernetes disposant d'autorisations suffisantes pour accéder aux objets Kubernetes de votre cluster Amazon EKS.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn’t have a resource type "svc"`

Cela peut être dû à l'une des raisons suivantes :

- Le cluster a été créé avec les informations d'identification d'un principal IAM et `kubectl` est configuré pour utiliser les informations d'identification d'un autre principal IAM. Pour résoudre ce problème, mettez à jour votre fichier `kube config` afin d’utiliser les informations d’identification à

l'origine de la création du cluster. Pour de plus amples informations, veuillez consulter the section called "Cluster d'accès avec kubectl".

- Si votre cluster répond aux exigences de plate-forme minimales indiquées dans la section Conditions préalables de la section Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS, aucune entrée d'accès n'existe auprès de votre principal IAM. S'il existe, les noms de groupes Kubernetes nécessaires ne lui sont pas définis ou la politique d'accès appropriée n'y est pas associée. Pour de plus amples informations, veuillez consulter the section called "Accorder des autorisations".
- Si votre cluster ne répond pas aux exigences minimales de plate-forme énoncées dans Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS, aucune entrée avec votre principal IAM n'existe dans le. `aws-auth` `ConfigMap` S'il existe, il n'est pas mappé aux noms de groupes Kubernetes liés à un `Role` Kubernetes ou dotés des autorisations nécessaires. `ClusterRole` Pour plus d'informations sur les objets d'autorisation basée sur les rôles (RBAC) Kubernetes, consultez la section Utilisation de l'autorisation RBAC dans la documentation de Kubernetes. Vous pouvez afficher vos `aws-auth` `ConfigMap` entrées actuelles *`my-cluster`* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :`eksctl get iamidentitymapping --cluster` *`my-cluster`* . Si aucune entrée contenant l'ARN de votre principal IAM ne figure dans le`ConfigMap`, entrez `eksctl create iamidentitymapping --help` dans votre terminal pour savoir comment en créer une.

Si vous installez et configurez la AWS CLI, vous pouvez configurer les informations d'identification IAM que vous utilisez. Pour plus d'informations, consultez la section Configuration de la AWS CLI dans le guide de l'utilisateur de l'interface de ligne de AWS commande. Vous pouvez également configurer `kubectl` pour utiliser un rôle IAM, si vous assumez un rôle IAM pour accéder aux objets Kubernetes de votre cluster. Pour de plus amples informations, veuillez consulter the section called "Cluster d'accès avec kubectl".

`hostname doesn't match`

La version Python de votre système doit être `2.7.9` ou ultérieure. Dans le cas contraire, vous recevrez `hostname doesn't match` des erreurs lors des appels AWS CLI vers Amazon EKS. Pour plus d'informations, voir Quelles sont les erreurs « le nom d'hôte ne correspond pas » ? dans les questions fréquemment posées sur les requêtes Python.

getsockopt: no route to host

Docker s'exécute dans la plage d'adresses CIDR `172.17.0.0/16` dans les clusters Amazon EKS. Nous recommandons que les sous-réseaux VPC de votre cluster ne chevauchent pas cette plage. Sinon, vous recevrez l'erreur suivante :

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

Si le message d'erreur s'affiche `Instances failed to join the Kubernetes cluster` dans le AWS Management Console, assurez-vous que l'accès au point de terminaison privé du cluster est activé ou que vous avez correctement configuré les blocs CIDR pour l'accès au point de terminaison public. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

Codes d'erreurs liées aux groupes de nœuds gérés

Si votre groupe de nœuds gérés rencontre un problème d'intégrité matérielle, Amazon EKS renvoie un code d'erreur pour vous aider à diagnostiquer le problème. Ces bilans de santé ne détectent pas les problèmes logiciels car ils sont basés sur les bilans de EC2 santé d'Amazon. La liste suivante décrit les codes d'erreur.

AccessDenied

Amazon EKS ou un ou plusieurs de vos nœuds gérés ne parviennent pas à s'authentifier ou à autoriser auprès de votre serveur d'API de cluster Kubernetes. Pour de plus amples informations sur la résolution d'une cause courante, consultez the section called "Correction d'une cause courante d'erreurs `AccessDenied` pour les groupes de nœuds gérés". AMIs Les fenêtres privées peuvent également provoquer ce code d'erreur en même temps que le message `Not authorized for images` d'erreur. Pour de plus amples informations, veuillez consulter the section called "`Not authorized for images`".

AmiIdNotFound

Nous n'avons pas trouvé l'ID AMI associé à votre modèle de lancement. Assurez-vous que l'AMI existe et est partagée avec votre compte.

AutoScalingGroupNotFound

Nous n'avons pas trouvé le groupe Auto Scaling associé au groupe de nœuds gérés. Vous pouvez peut-être recréer un groupe Auto Scaling avec les mêmes paramètres pour effectuer une récupération.

ClusterUnreachable

Amazon EKS ou un ou plusieurs de vos nœuds gérés ne peuvent pas communiquer avec le serveur API de votre cluster Kubernetes. Cela peut se produire s'il y a des interruptions de réseau ou si les serveurs d'API temporisent le traitement des demandes.

Eco 2 SecurityGroupNotFound

Nous n'avons pas pu trouver le groupe de sécurité du cluster pour le cluster. Vous devez recréer votre cluster.

Eco 2 SecurityGroupDeletionFailure

Nous n'avons pas pu supprimer le groupe de sécurité d'accès à distance pour votre groupe de nœuds gérés. Supprimez toutes les dépendances du groupe de sécurité.

Eco 2 LaunchTemplateNotFound

Nous n'avons pas trouvé le modèle de EC2 lancement Amazon pour votre groupe de nœuds géré. Vous devez recréer votre groupe de nœuds pour effectuer une récupération.

Eco 2 LaunchTemplateVersionMismatch

La version du modèle de EC2 lancement Amazon pour votre groupe de nœuds gérés ne correspond pas à la version créée par Amazon EKS. Vous pouvez peut-être revenir à la version Amazon EKS créée pour effectuer une récupération.

IamInstanceProfileNotFound

Nous n'avons pas trouvé le profil d'instance IAM pour votre groupe de nœuds gérés. Vous pouvez peut-être recréer un profil d'instance avec les mêmes paramètres pour effectuer une récupération.

IamNodeRoleNotFound

Nous n'avons pas trouvé le rôle IAM pour votre groupe de nœuds gérés. Vous pouvez peut-être recréer un rôle IAM avec les mêmes paramètres pour effectuer une récupération.

AsgInstanceLaunchFailures

Votre groupe Auto Scaling rencontre des problèmes lors d'une tentative de lancement d'instances.

NodeCreationFailure

Vos instances lancées ne peuvent pas s'enregistrer auprès de votre cluster Amazon EKS. Les causes courantes de cet échec sont des autorisations de rôle IAM de nœud insuffisantes ou l'absence d'un accès Internet sortant pour les nœuds. Vos nœuds doivent répondre à l'une des exigences suivantes :

- Accès à Internet à l'aide d'une adresse IP publique. Le groupe de sécurité associé au sous-réseau dans lequel se trouve le nœud doit autoriser la communication. Pour plus d'informations, consultez the section called "Exigences et considérations requises pour les sous-réseaux" et the section called "Exigences de groupe de sécurité".
- Vos nœuds et VPC doivent répondre aux exigences de la section Déployer des clusters privés avec un accès Internet limité.

InstanceLimitExceeded

Votre AWS compte n'est plus en mesure de lancer d'autres instances du type d'instance spécifié. Vous pouvez peut-être demander une augmentation de la limite d' EC2 instance Amazon pour récupérer.

InsufficientFreeAddresses

Un ou plusieurs sous-réseaux associés à votre groupe de nœuds gérés ne disposent pas d'un nombre suffisant d'adresses IP disponibles pour les nouveaux nœuds.

InternalFailure

Ces erreurs sont généralement dues à un problème côté serveur Amazon EKS.

Correction d'une cause courante d'erreurs **`AccessDenied`** pour les groupes de nœuds gérés

La cause la plus courante d'erreurs `AccessDenied` lors de la réalisation d'opérations sur des groupes de nœuds gérés est l'absence de `eks:node-manager` `ClusterRole` ou `ClusterRoleBinding`. Amazon EKS configure ces ressources dans votre cluster dans le cadre de l'onboarding avec les groupes de nœuds gérés, et celles-ci sont nécessaires pour gérer les groupes de nœuds.

La `ClusterRole` peut changer au fil du temps, mais elle doit ressembler à l'exemple suivant :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

La `ClusterRoleBinding` peut changer au fil du temps, mais elle doit ressembler à l'exemple suivant :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Vérifiez que la `eks:node-manager` `ClusterRole` existe.

```
kubectl describe clusterrole eks:node-manager
```

Si elle est présente, comparez la sortie à l'exemple de la `ClusterRole`précédente.

Vérifiez que la `eks:node-manager` `ClusterRoleBinding` existe.

```
kubectl describe clusterrolebinding eks:node-manager
```

Si elle est présente, comparez la sortie à l'exemple de la `ClusterRoleBinding`précédente.

Si vous avez identifié un élément manquant, défectueux `ClusterRole` ou à `ClusterRoleBinding` l'origine d'une `AcessDenied` erreur lors de la demande d'opérations de groupe de nœuds gérés, vous pouvez le restaurer. Enregistrez le contenu suivant dans un fichier nommé *`eks-node-manager-role.yaml`*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
```

```
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Appliquez le fichier.

```
kubectl apply -f eks-node-manager-role.yaml
```

Réessayez l'opération de groupe de nœuds pour voir si cela a résolu votre problème.

`Not authorized for images`

L'une des causes potentielles d'un message `Not authorized for images` d'erreur est l'utilisation d'une AMI Windows Amazon EKS privée pour lancer des groupes de nœuds gérés par Windows. Après la sortie de nouveaux Windows AMIs, AWS AMIs les versions datant de plus de 4 mois deviennent privées, ce qui les rend inaccessibles. Si votre groupe de nœuds gérés utilise une AMI Windows privée, pensez à mettre à jour votre groupe de nœuds gérés Windows. Bien que nous ne puissions pas garantir que nous puissions fournir un accès à AMIs ce qui a été rendu privé, vous pouvez demander l'accès en déposant un ticket auprès du AWS Support. Pour plus d'informations, consultez la section Patches du guide de EC2 l'utilisateur Amazon.

Le nœud est en **`NotReady`** état

Si votre nœud entre dans un `NotReady` état, cela indique probablement qu'il n'est pas en bon état et qu'il n'est pas disponible pour planifier de nouveaux pods. Cela peut se produire pour diverses

raisons, telles que le fait que le nœud ne dispose pas de ressources suffisantes pour le processeur, la mémoire ou l'espace disque disponible.

Pour Windows optimisé pour Amazon EKS AMIs, aucune réservation n'est prévue pour les ressources de calcul spécifiées par défaut dans la `kubelet` configuration. Pour éviter les problèmes de ressources, vous pouvez réserver des ressources de calcul pour les processus du système en `kubelet` fournissant des valeurs de configuration pour Kube-reserved system-reserved. and/or Pour ce faire, utilisez le paramètre de `-KubeletExtraArgs` ligne de commande du script bootstrap. Pour plus d'informations, consultez la section Reserve Compute Resources for System Daemons dans la documentation Kubernetes et the section called "Paramètres de configuration du script d'amorçage" dans ce guide de l'utilisateur.

Collecteur de journaux EKS

Pour résoudre les problèmes liés aux nœuds Amazon EKS, un script prédéfini est disponible sur les nœuds situés à l'adresse. `/etc/eks/log-collector-script/eks-log-collector.sh` Vous pouvez utiliser le script pour collecter des journaux de diagnostic pour les cas de support et le dépannage général.

Utilisez la commande suivante pour exécuter le script sur votre nœud :

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

> **Note**
>
> Si le script n'est pas présent à cet emplacement. Vous pouvez manuellement télécharger et exécuter le script à l'aide de la commande suivante :
>
> ```
> curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/
> eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

Le script collecte les informations de diagnostic suivantes.

```
$ sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

```
      This is version 0.7.8. New versions can be found at https://github.com/awslabs/
amazon-eks-ami/blob/main/log-collector-script/

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
...
...

 Done... your bundled logs are located in /var/log/eks_i-EXAMPLE_2025-03-25_0000-
UTC_0.7.8.tar.gz
```

Les informations de diagnostic sont collectées et stockés dans :

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

Pour récupérer le bundle de journaux pour les nœuds Bottlerocket, reportez-vous à Bottlerocket Log pour plus de détails.

Le réseau d'exécution du conteneur n'est pas prêt

Vous pouvez recevoir une erreur `Container runtime network not ready` et des erreurs d'autorisation similaires aux suivantes :

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

Les raisons possibles sont les suivantes :

1. Soit vous n'en avez pas `aws-auth` `ConfigMap` sur votre cluster, soit celui-ci n'inclut pas d'entrées pour le rôle IAM avec lequel vous avez configuré vos nœuds.

   Pour résoudre le problème, consultez les entrées existantes dans votre *`my-cluster`* en remplaçant la commande suivante `ConfigMap` par le nom de votre cluster, puis en exécutant

la commande modifiée :`eksctl get iamidentitymapping --cluster` *`my-cluster`* . Si vous recevez un message d'erreur de la part de la commande, cela peut être dû au fait que votre cluster ne possède pas de `aws-authConfigMap`. La commande suivante ajoute une entrée à la `ConfigMap`. Si le `ConfigMap` n'existe pas, la commande le crée également. *`111122223333`*Remplacez-le par l'ID de AWS compte du rôle IAM et *`myAmazonEKSNodeRole`* par le nom du rôle de votre nœud.

```
eksctl create iamidentitymapping --cluster my-cluster \
    --arn arn:aws: iam::111122223333:role/myAmazonEKSNodeRole --group
 system:bootstrappers,system:nodes \
    --username system:node:{{EC2PrivateDNSName}}
```

L'ARN du rôle que vous spécifiez ne peut pas inclure de chemin autre que`/`. Par exemple, si le nom de votre rôle est`development/apps/my-role`, vous devrez le remplacer `my-role` lorsque vous spécifiez l'ARN du rôle. Assurez-vous que vous spécifiez l'ARN pour le rôle IAM du nœud (et non l'ARN du profil d'instance).

2. Vos nœuds autogérés se trouvent dans un cluster dont la version de plate-forme est la version minimale répertoriée dans les conditions requises dans la rubrique Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS, mais aucune entrée n'est répertoriée dans le `aws-auth ConfigMap` (voir point précédent) pour le rôle IAM du nœud ou aucune entrée d'accès n'existe pour le rôle. Pour résoudre le problème, consultez vos entrées d'accès existantes *`my-cluster`* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :`aws eks list-access-entries --cluster-name` *`my-cluster`* . La commande suivante ajoute une entrée d'accès pour le rôle IAM du nœud. *`111122223333`*Remplacez-le par l'ID de AWS compte du rôle IAM et *`myAmazonEKSNodeRole`* par le nom du rôle de votre nœud. Si vous possédez un nœud Windows, remplacez-le *`EC2_LINUX`* par`EC2_Windows`. Assurez-vous que vous spécifiez l'ARN pour le rôle IAM du nœud (et non l'ARN du profil d'instance).

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/myAmazonEKSNodeRole --type EC2_LINUX
```

Délai d'expiration de la liaison TLS

Lorsqu'un nœud est incapable d'établir une connexion avec le point de terminaison du serveur d'API public, vous pouvez voir une erreur similaire à l'erreur suivante.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

Le processus `kubelet` se reproduira continuellement et testera le point de terminaison du serveur d'API. L'erreur peut également se produire temporairement au cours de toute procédure qui effectue une mise à jour continue du cluster dans le plan de contrôle, telle qu'une modification de configuration ou une mise à jour de version.

Pour résoudre le problème, vérifiez la table de routage et les groupes de sécurité pour vous assurer que le trafic provenant des nœuds peut atteindre le point de terminaison public.

InvalidClientTokenId

Si vous utilisez des rôles IAM pour les comptes de service d'un pod ou si vous êtes DaemonSet déployé sur un cluster dans une AWS région de Chine, et que vous n'avez pas défini la variable d'`AWS_DEFAULT_REGIO`Nenvironnement dans la spécification, le pod ou le pod DaemonSet peut recevoir le message d'erreur suivant :

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

Pour résoudre le problème, vous devez ajouter la variable d'`AWS_DEFAULT_REGIO`Nenvironnement à votre Pod ou à votre DaemonSet spécification, comme indiqué dans l'exemple de spécification de Pod suivant.

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

Les groupes de nœuds doivent correspondre à la version de Kubernetes avant de mettre à niveau le plan de contrôle

Avant de mettre à niveau un plan de contrôle vers une nouvelle version de Kubernetes, la version mineure des nœuds gérés et Fargate de votre cluster doit être identique à la version actuelle de votre plan de contrôle. L'API `update-cluster-version` Amazon EKS rejette les demandes tant que vous n'avez pas mis à niveau tous les nœuds gérés par Amazon EKS vers la version actuelle du cluster. Amazon EKS permet APIs de mettre à niveau les nœuds gérés. Pour plus d'informations sur la mise à niveau de la version Kubernetes d'un groupe de nœuds gérés, consultez. the section called "Mettre à jour" Pour mettre à niveau la version d'un nœud Fargate, supprimez le pod représenté par le nœud et redéployez le pod après avoir mis à niveau votre plan de contrôle. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour la version de Kubernetes".

Lors du lancement de nombreux nœuds, des erreurs **`Too Many Requests`** se produisent

Si vous lancez plusieurs nœuds simultanément, un message d'erreur peut s'afficher dans les journaux d'exécution des données EC2 utilisateur d'Amazon indiquant que : `Too Many Requests` Cela peut se produire parce que le plan de contrôle est surchargé d'appels `describeCluster`. Cette surcharge se traduit par une limitation, des nœuds qui ne parviennent pas à exécuter le script d'amorçage et des nœuds qui ne parviennent pas à rejoindre le cluster.

Assurez-vous que les `--dns-cluster-ip` arguments `--apiserver-endpoint--b64-cluster-ca`, et sont transmis au script bootstrap du nœud. Lorsque vous incluez ces arguments, le script bootstrap n'a pas besoin d'effectuer un `describeCluster` appel, ce qui permet d'éviter une surcharge du plan de contrôle. Pour de plus amples informations, veuillez consulter the section called "Fournissez des données utilisateur pour transmettre des arguments au `bootstrap.sh` fichier inclus dans une Linux/Bottlerocket AMI optimisée pour Amazon EKS".

Réponse d'erreur non autorisée HTTP 401 sur les requêtes du serveur API Kubernetes

Ces erreurs s'affichent si le jeton de compte de service d'un Pod a expiré sur un cluster.

Le serveur d'API Kubernetes de votre cluster Amazon EKS rejette les demandes contenant des jetons datant de plus de 90 jours. Dans les versions précédentes de Kubernetes, les jetons n'avaient

pas de période d'expiration. Cela signifie que les clients qui s'appuient sur ces jetons doivent les actualiser dans l'heure. Pour empêcher le serveur d'API Kubernetes de rejeter votre demande en raison d'un jeton non valide, la version du SDK client Kubernetes utilisée par votre charge de travail doit être identique ou ultérieure aux versions suivantes :

- Go version `0.15.7` et ultérieure
- Python version `12.0.0` et ultérieure
- Java version `9.0.0` et ultérieure
- JavaScript version `0.10.3` et versions ultérieures
- Branche `master` Ruby
- Haskell version `0.3.0.0`
- Version C# `7.0.5` et versions ultérieures

Vous pouvez identifier tous les pods existants de votre cluster qui utilisent des jetons périmés. Pour de plus amples informations, veuillez consulter the section called "Jetons de compte de service".

La version de la plateforme Amazon EKS est inférieure de plus de deux versions à la version actuelle de la plateforme

Cela peut se produire lorsqu'Amazon EKS n'est pas en mesure de mettre à jour automatiquement le eks/latest/userguide/platform fichier -versions.html [platform-version, type="documentation "] de votre cluster. Bien qu'il existe de nombreuses causes à cela, certaines des causes les plus courantes suivent. Si l'un de ces problèmes s'applique à votre cluster, il est possible qu'il fonctionne toujours, mais la version de sa plateforme ne sera tout simplement pas mise à jour par Amazon EKS.

Problème

Le Rôle IAM du cluster a été supprimé : ce rôle a été spécifié lors de la création du cluster. Vous pouvez voir quel rôle a été spécifié à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

L'exemple qui suit illustre un résultat.

```
eksClusterRole
```

Solution

Création d'un nouveau rôle IAM du cluster portant le même nom.

Problème

Un sous-réseau spécifié lors de la création du cluster a été supprimé : les sous-réseaux à utiliser avec le cluster ont été spécifiés lors de la création du cluster. Vous pouvez voir quels sous-réseaux ont été spécifiés à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

L'exemple qui suit illustre un résultat.

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Solution

Vérifiez si le sous-réseau IDs existe dans votre compte.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

L'exemple qui suit illustre un résultat.

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

Si le sous-réseau IDs renvoyé dans la sortie ne correspond pas au sous-réseau spécifié lors de la création du cluster, si vous souhaitez IDs qu'Amazon EKS mette à jour le cluster, vous devez modifier les sous-réseaux utilisés par le cluster. En effet, si vous avez spécifié plus de deux sous-réseaux

lors de la création de votre cluster, Amazon EKS sélectionne de manière aléatoire les sous-réseaux que vous avez spécifiés pour y créer de nouvelles interfaces réseau Elastic. Ces interfaces réseau permettent au plan de contrôle de communiquer avec vos nœuds. Amazon EKS ne mettra pas à jour le cluster si le sous-réseau sélectionné n'existe pas. Vous n'avez aucun contrôle sur les sous-réseaux que vous avez spécifiés lors de la création du cluster dans lesquels Amazon EKS choisit de créer une nouvelle interface réseau.

Lorsque vous lancez une mise à jour de version de Kubernetes pour votre cluster, la mise à jour peut échouer pour la même raison.

Problème

Un groupe de sécurité spécifié lors de la création du cluster a été supprimé : si vous avez spécifié des groupes de sécurité lors de la création du cluster, vous pouvez les voir à l' IDs aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

L'exemple qui suit illustre un résultat.

```
[
    "sg-EXAMPLE1"
]
```

Si elle `[]` est renvoyée, aucun groupe de sécurité n'a été spécifié lors de la création du cluster et aucun groupe de sécurité manquant n'est à l'origine du problème. Si des groupes de sécurité sont renvoyés, vérifiez qu'ils existent dans votre compte.

Solution

Vérifiez si ces groupes de sécurité existent dans votre compte.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

L'exemple qui suit illustre un résultat.

```
[
```

```
"sg-EXAMPLE2"
]
```

Si le groupe de sécurité IDs renvoyé dans la sortie ne correspond pas au groupe de sécurité spécifié lors de la création du cluster, si vous souhaitez IDs qu'Amazon EKS mette à jour le cluster, vous devez modifier les groupes de sécurité utilisés par le cluster. Amazon EKS ne met pas à jour un cluster si le groupe de sécurité IDs spécifié lors de la création du cluster n'existe pas.

Lorsque vous lancez une mise à jour de version de Kubernetes pour votre cluster, la mise à jour peut échouer pour la même raison.

- Vous n'avez pas au moins six (mais nous recommandons 16) adresses IP disponibles dans chacun des sous-réseaux que vous avez spécifiés lors de la création de votre cluster. Si vous n'avez pas suffisamment d'adresses IP disponibles dans le sous-réseau, vous devez soit libérer des adresses IP dans le sous-réseau, soit modifier les sous-réseaux utilisés par le cluster pour utiliser des sous-réseaux avec suffisamment d'adresses IP disponibles.
- Vous avez activé le chiffrement secret lorsque vous avez créé votre cluster et la clé AWS KMS que vous avez spécifiée a été supprimée. Si vous souhaitez qu'Amazon EKS mette à jour le cluster, vous devez créer un nouveau cluster

Codes d'erreur FAQs et d'intégrité du cluster avec chemins de résolution

Amazon EKS détecte les problèmes liés à vos clusters EKS et à l'infrastructure du cluster et les stocke dans l'objet d'état de la ressource de votre cluster EKS. Vous pouvez détecter et résoudre les problèmes de cluster plus rapidement à l'aide des informations relatives à l'état du cluster. Cela vous permet de créer des environnements d'applications plus sécurisés et up-to-date. En outre, il peut s'avérer impossible pour vous de passer à des versions plus récentes de Kubernetes ou pour Amazon EKS d'installer des mises à jour de sécurité sur un cluster dégradé en raison de problèmes liés à l'infrastructure ou à la configuration du cluster nécessaires. Amazon EKS peut mettre 3 heures pour détecter les problèmes ou détecter qu'un problème est résolu.

La santé d'un cluster Amazon EKS est une responsabilité partagée entre Amazon EKS et ses utilisateurs. Vous êtes responsable de l'infrastructure préalable des rôles IAM et des sous-réseaux Amazon VPC, ainsi que des autres infrastructures nécessaires, qui doivent être fournies à l'avance. Amazon EKS détecte les modifications apportées à la configuration de cette infrastructure et du cluster.

Pour accéder à l'état de santé de votre cluster dans la console Amazon EKS, recherchez un tableau intitulé Problèmes de santé dans l'onglet Problèmes de santé du cluster du tableau de bord d'observabilité accessible depuis la page détaillée du cluster Amazon EKS. Ces données seront également disponibles en appelant l'`DescribeCluster`action dans l'API EKS, par exemple depuis l'interface de ligne de AWS commande.

Pourquoi utiliser cette fonctionnalité ?

Vous bénéficierez d'une visibilité accrue sur l'état de santé de votre cluster Amazon EKS, diagnostiquerez et résoudrez rapidement les problèmes, sans avoir à passer du temps à déboguer ou à ouvrir des dossiers de AWS support. Par exemple : vous avez accidentellement supprimé un sous-réseau pour le cluster Amazon EKS, Amazon EKS ne sera pas en mesure de créer des interfaces réseau entre comptes et des commandes de la AWS CLI Kubernetes `kubectl` telles que exec ou logs. `kubectl` L'erreur suivante s'affiche : `Error from server: error dialing backend: remote error: tls: internal error.` Le problème d'état Amazon EKS indique : `subnet-da60e280 was deleted: could not create network interface`.

Comment cette fonctionnalité est-elle liée ou fonctionne-t-elle avec d'autres AWS services ?

Les rôles IAM et les sous-réseaux Amazon VPC sont deux exemples d'infrastructure préalable avec laquelle l'état du cluster détecte les problèmes. Cette fonctionnalité renvoie des informations détaillées si ces ressources ne sont pas correctement configurées.

Un cluster présentant des problèmes de santé est-il payant ?

Oui, chaque cluster Amazon EKS est facturé au tarif standard d'Amazon EKS. La fonctionnalité liée à l'état du cluster est disponible sans frais supplémentaires.

Cette fonctionnalité fonctionne-t-elle avec les clusters Amazon EKS sur AWS Outposts ?

Oui, des problèmes de cluster sont détectés pour les clusters EKS dans le AWS cloud, y compris les clusters étendus sur les AWS Outposts et les clusters locaux sur les AWS Outposts. L'état du cluster ne détecte pas les problèmes liés à Amazon EKS Anywhere ou Amazon EKS Distro (EKS-D).

Puis-je être averti lorsque de nouveaux problèmes sont détectés ?

Oui AWS envoie un e-mail et une notification au Personal Health Dashboard lorsque de nouveaux problèmes de santé du cluster sont détectés.

La console m'avertit-elle en cas de problème de santé ?

Oui, tout cluster présentant des problèmes d'état présentera une bannière en haut de la console.

Les deux premières colonnes sont celles qui sont nécessaires pour les valeurs de réponse de l'API. Le troisième champ de l' ClusterIssueobjet Health est ResourceIds, dont le retour dépend du type de problème.

Code	Message	ResourceIds	Le cluster est-il récupérable ?
SOUS-RÉSEAU NON TROUVÉ	Nous n'avons pas pu trouver un ou plusieurs sous-rése aux actuellement associés à votre cluster. Appelez l'API update-cluster-con fig Amazon EKS pour mettre à jour les sous-réseaux.	ID de sous-réseaux	Oui
GROUPE_DE SÉCURITÉ INTROUVABLE	Nous n'avons pas pu trouver un ou plusieurs groupes de sécurité actuellem ent associés à votre cluster. Appelez l' update-cluster-con figAPI Amazon EKS pour mettre à jour les groupes de sécurité	ID de groupe de sécurité	Oui
IP NON DISPONIBLE	Un ou plusieurs sous-réseaux associés à votre cluster ne disposent pas d'un nombre suffisant d'adresses IP pour qu'Amazon EKS puisse effectuer des	ID de sous-réseaux	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
	opérations de gestion de cluster. Libérez des adresses dans le ou les sous-réseaux ou associez différents sous-réseaux à votre cluster à l'aide de l'API Amazon EKS update-cluster-config.		
VPC_INTROUVABLE	Nous n'avons pas trouvé le VPC associé à votre cluster. Vous devez supprimer et recréer votre cluster.	ID du VPC	Non
ASSUME_ROLE_ACCESS_DENIED	Votre cluster n'utilise pas Amazon EKS service-linked-role. Nous ne pouvions pas assumer le rôle associé à votre cluster pour effectuer les opérations de gestion Amazon EKS requises. Vérifiez que le rôle existe et qu'il dispose de la politique de confiance requise.	Rôle IAM du cluster	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
AUTORISATION D'ACCÈS REFUSÉE	Votre cluster n'utilise pas Amazon EKS service-linked-role. Le rôle associé à votre cluster n'accorde pas les autorisat ions suffisantes à Amazon EKS pour effectuer les opération s de gestion requises. Vérifiez les politique s associées au rôle de cluster et si des politiques de refus distinctes sont appliquées.	Rôle IAM du cluster	Oui
ASSUME_RO LE_ACCESS _DENIED_U SING_SLR	Nous ne pouvions pas assumer la gestion du cluster Amazon EKS service-linked-role. Vérifiez que le rôle existe et qu'il dispose de la politique de confiance requise.	L'Amazon EKS service-linked-role	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
AUTORISATION D'ACCÈS REFUSÉE À L'AIDE DE SLR	La gestion du cluster Amazon EKS service-linked-role n'accorde pas d'autorisations suffisantes à Amazon EKS pour effectuer les opérations de gestion requises. Vérifiez les politique s associées au rôle de cluster et si des politiques de refus distinctes sont appliquées.	L'Amazon EKS service-linked-role	Oui
OPT_IN_OB LIGATOIRE	Aucun abonnemen t au EC2 service Amazon n'est associé à votre compte. Mettez à jour les abonnements de votre compte sur la page des paramètres de votre compte.	N/A	Oui
STS_REGIO NAL_ENDPO INT_DISABLED	Le point de terminais on régional STS est désactivé. Activez le point de terminaison pour qu'Amazon EKS effectue les opération s de gestion de cluster requises.	N/A	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
KMS_KEY_DISABLED	La clé AWS KMS associée à votre cluster est désactivée. Réactivez la clé pour récupérer votre cluster.	Le porte-clés KMS	Oui
KMS_KEY_NOT_FOUND	Nous n'avons pas trouvé la clé AWS KMS associée à votre cluster. Vous devez supprimer et recréer le cluster.	L'ARN de la clé KMS	Non
KMS_GRANT_RÉVOQUÉ	Les autorisations pour la clé AWS KMS associée à votre cluster sont révoquées. Vous devez supprimer et recréer le cluster.	Le porte-clés KMS	Non

Étendez les fonctionnalités d'Amazon EKS avec des projets open source

Ces projets open source étendent les fonctionnalités des clusters Kubernetes exécutés sur ou en dehors de ceux-ci AWS, y compris les clusters gérés par Amazon EKS.

Support pour les logiciels déployés sur EKS

Lorsque vous consultez la documentation Amazon EKS, vous trouverez des références à divers outils et logiciels open source dans nos procédures et exemples. Ces outils incluent le serveur de métriques Kubernetes et le gestionnaire de certificats.

Veuillez noter que tout logiciel tiers ou open source que vous choisissez de déployer ne relève pas du champ d'application de vos Contrats de AWS Support. L'un des avantages de Kubernetes est la communauté open source active. Nous vous recommandons de travailler directement avec les communautés open source et les responsables de projets concernés afin d'établir des canaux de support appropriés pour ces composants. Pour plus d'informations, consultez les projets diplômés et en incubation associés à la Cloud Native Computing Foundation (CNCF).

L'écosystème Kubernetes comprend de nombreux projets et composants qui s'accompagnent de différents niveaux de soutien communautaire, de temps de réponse et de cas d'utilisation prévus. Lorsque vous implémentez ces technologies avec EKS, assurez-vous de bien comprendre la matrice de support de chaque composant.

AWS gère les composants open source que nous intégrons dans le plan de contrôle EKS. Cela inclut notre pipeline de sécurité complet couvrant la vérification des versions, l'analyse des vulnérabilités, les tests de validation et la gestion des correctifs pour toutes les images de conteneurs et les fichiers binaires que nous distribuons. Par exemple, AWS est responsable du serveur d'API Kubernetes. Le serveur d'API Kubernetes est couvert par le contrat de niveau de service Amazon EKS. Vous pouvez utiliser votre plan de support Amazon Web Services pour résoudre les problèmes liés au serveur d'API Kubernetes ou obtenir des conseils généraux.

Vous devez examiner attentivement le support proposé pour les différents modules complémentaires Amazon EKS. AWS les modules complémentaires sont les seuls types de modules complémentaires Amazon EKS entièrement pris en charge par AWS. AWS Les modules complémentaires Marketplace sont principalement pris en charge par AWS les partenaires. Les modules complémentaires

communautaires bénéficient d'une assistance de base sur le cycle de vie de AWS. Pour plus d'informations, consultez la section Support des modules complémentaires.

Tous les modules complémentaires EKS, quel que soit leur type, bénéficient d'un support de base sur le cycle de vie de la part d'EKS, y compris les modules complémentaires Marketplace. Le support de base relatif au cycle de vie inclut l'installation et la désinstallation du module complémentaire. Pour plus d'informations sur les types de modules complémentaires Amazon EKS disponibles et les niveaux de support associés, consultez la section Étendue du support pour les modules complémentaires Amazon EKS. Pour voir les modules complémentaires entièrement pris en charge par AWS, consultez les modules complémentaires Amazon Web Services.

- Pour plus d'informations sur nos pratiques de sécurité et les limites de support, consultez la section Sécurité dans Amazon EKS.
- Pour plus d'informations sur les modules complémentaires destinés à la communauté et au AWS marché disponibles via les modules complémentaires Amazon EKS, consultez la section Support des modules complémentaires EKS.

Outils de gestion

Outils de gestion connexes pour les clusters Amazon EKS et Kubernetes.

eksctl

`eksctl` est un outil de CLI pour créer des clusters sur Amazon EKS.

- URL du projet
- Documentation du projet
- AWS blog open source : eksctl : cluster Amazon EKS avec une seule commande

AWS contrôleurs pour Kubernetes

Avec AWS Controllers for Kubernetes, vous pouvez créer et gérer des AWS ressources directement à partir de votre cluster Kubernetes.

- URL du projet
- AWS blog open source : l'opérateur AWS de service pour Kubernetes est désormais disponible

Flux CD

Flux est un outil que vous pouvez utiliser pour gérer la configuration de votre cluster à l'aide de Git. Il utilise un opérateur dans le cluster pour déclencher des déploiements à l'intérieur de Kubernetes. Pour plus d'informations sur les opérateurs, consultez OperatorHub.io on GitHub.

- URL du projet
- Documentation du projet

CDK pour Kubernetes

Avec le CDK pour Kubernetes (cdk8s), vous pouvez définir des applications et des composants Kubernetes en utilisant des langages de programmation familiers. Les applications cdk8s se synthétisent en manifestes Kubernetes standard, qui peuvent être appliqués à n'importe quel cluster Kubernetes.

- URL du projet
- Documentation du projet
- AWS blog sur les conteneurs : Présentation de cdk8s+ : piloté par l'intention pour les objets Kubernetes APIs

Réseaux

Projets de mise en réseau connexes pour les clusters Amazon EKS et Kubernetes.

Plugin CNI Amazon VPC pour Kubernetes

Amazon EKS prend en charge la mise en réseau VPC via le plugin CNI Amazon VPC pour Kubernetes. Le plugin attribue une adresse IP de votre VPC à chaque Pod.

- URL du projet
- Documentation du projet

AWS Contrôleur Load Balancer pour Kubernetes

Le AWS Load Balancer Controller permet de gérer les AWS Elastic Load Balancers pour un cluster Kubernetes. Il satisfait les ressources Kubernetes Ingress en provisionnant des équilibreurs de

charge d'application. AWS Il satisfait les ressources du service Kubernetes en fournissant des équilibreurs de charge réseau. AWS

- URL du projet
- Documentation du projet

ExternalDNS

ExternalDNS synchronise les services Kubernetes exposés et les entrées avec les fournisseurs DNS tels qu'Amazon Route 53 et Service Discovery. AWS

- URL du projet
- Documentation du projet

Machine learning

Projets de machine learning connexes pour les clusters Amazon EKS et Kubernetes.

Kubeflow

Boîte à outils d'apprentissage automatique pour Kubernetes.

- URL du projet
- Documentation du projet
- AWS blog open source : Kubeflow sur Amazon EKS

Auto Scaling

Projets de scalabilité automatique connexes pour les clusters Amazon EKS et Kubernetes.

Cluster Autoscaler

Cluster Autoscaler est un outil qui ajuste automatiquement la taille du cluster Kubernetes en fonction de la sollicitation de l'UC et de la mémoire.

- URL du projet
- Documentation du projet

- Atelier Amazon EKS : Cluster Autoscaler

Karpenter

Karpenter est un autoscaler de nœuds Kubernetes conçu pour la flexibilité, les performances et la simplicité.

- URL du projet
- Documentation du projet
- Atelier Amazon EKS : Karpenter

Escalator

Escalator est un outil de scalabilité automatique horizontal optimisé de lot ou de tâche pour Kubernetes.

- URL du projet
- Documentation du projet

Surveillance

Projets de surveillance connexes pour les clusters Amazon EKS et Kubernetes.

Prometheus

Prometheus est une boîte à outils de surveillance de systèmes et d'alerte open source.

- URL du projet
- Documentation du projet
- Atelier Amazon EKS : https://eksworkshop.com/intermediate/240_monitoring/

Intégration continue/déploiement continu

Projets d'intégration/de déploiement en continu (CI/CD) connexes pour les clusters Amazon EKS et Kubernetes.

Jenkins X

Solution CI/CD pour les applications cloud modernes sur les clusters Amazon EKS et Kubernetes.

- URL du projet
- Documentation du projet

En savoir plus sur les nouvelles fonctionnalités et la feuille de route d'Amazon EKS

Vous pouvez en savoir plus sur les nouvelles fonctionnalités d'Amazon EKS en accédant au fil Nouveautés de la page Nouveautés. AWS Vous pouvez également consulter la feuille de route GitHub, qui vous informe des fonctionnalités et priorités à venir afin de planifier la manière dont vous souhaitez utiliser Amazon EKS à l'avenir. Vous pouvez nous fournir des commentaires directs sur les priorités de la feuille de route.

Historique du document

Le tableau suivant décrit certaines des principales mises à jour et nouvelles fonctionnalités du guide de l'utilisateur Amazon EKS. Pour recevoir des notifications lorsque ce tableau reçoit une nouvelle entrée, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss
```

Modification	Description	Date
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes `1.33.2`, `1.32.6`, `1.31.10` et `1.30.14`.	30 juillet 2025
Fonction VPC CNI Multi-NIC pour les pods à plusieurs logements	Amazon EKS ajoute des pods multi-hôtes au VPC CNI. Vous pouvez désormais configurer une charge de travail et les adresses IP attribuées au VPC CNI entre chaque carte réseau de l' EC2 instance et chaque pod. L'application peut établir des connexions simultanées pour utiliser la bande passante de chaque carte réseau. Chaque interface réseau est configurée dans le même sous-réseau et les mêmes groupes de sécurité que le nœud. Auparavant, vous deviez utiliser Multus CNI pour	15 juillet 2025

	en exécuter plusieurs autres afin de créer des pods multi-hôtes. CNIs La documentation et les étapes à suivre pour ce faire ont été déplacées vers le eks/latest/userguide/podfichier-multus.html.	
Mise à jour du contenu de dépannage du VPC CNI	La page de résolution des problèmes relative à la politique réseau Kubernetes a été étendue dans le VPC CNI. Ajout des CRDs autorisations RBAC et de 12 problèmes connus.	30 juin 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif de Kubernetes dans ces versions de plateforme.	26 juin 2025
AWS mises à jour des politiques gérées	`ssmmessages:OpenDataChannel` Autorisation ajoutée pour`AmazonEKSLocalOutpostServiceRolePolicy` .	26 juin 2025

AWS mises à jour des politiques gérées	Des autorisations ont été ajoutées `AmazonEKSComputePolicy` pour autoriser le mode automatique Amazon EKS à lancer des instances en utilisant les réservations de capacité EC2 à la demande de votre compte. `AmazonEKSServiceRolePolicy` J'ai également ajouté les autorisations `AmazonEKSComputePolicy` permettant à Amazon EKS de créer le rôle lié au service EC2 Spot en votre nom.	20 juin 2025
Mises à jour des politiques gérées	`AmazonEKSDashboardConsoleReadOnly` Politique ajoutée.	19 juin 2025
Mise à jour du mode automatique d'Amazon EKS vers `NodeClass`	Le `NodeClass` modèle pour les nœuds en mode automatique a ajouté une configuration pour des sous-réseaux de pods distincts. Cela ajoute les clés facultatives `podSubnetSelectorTerms` et permet `podSecurityGroupSelectorTerms` de définir les sous-réseaux et les groupes de sécurité pour les pods.	13 juin 2025

Ciblez les rôles secondaires et multicomptes avec EKS Pod Identities	Amazon EKS ajoute des rôles IAM cibles à EKS Pod Identities pour automatiser le chaînage des rôles. Vous pouvez l'utiliser pour assumer automatiquement un rôle dans un autre compte et EKS Pod Identity alterne les informations d'identification temporaires. Chaque association Pod Identity doit d'abord avoir un rôle IAM dans le même compte à assumer, puis elle utilise ce rôle pour assumer le rôle cible.	11 juin 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif de Kubernetes dans ces versions de plateforme.	11 juin 2025
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Taipei`ap-east-2`) ().	6 juin 2025
`IPv6`contrôle d'accès pour les points de terminaison publics à double pile pour les nouveaux clusters `IPv6`		5 juin 2025

Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes`1.32.5`,, `1.31.9` et. `1.30.13`	30 mai 2025
Nouvelles informations sur les clusters pour les nœuds hybrides EKS	Amazon EKS ajoute de nouvelles informations sur les clusters qui vérifient la configuration de vos nœuds hybrides. Ces vérifications d'informations vous avertiron t des problèmes liés aux nœuds et aux pods sur site et à la configuration réseau à distance du cluster.	29 mai 2025
Version de Kubernetes `1.33`	Ajout de la prise en `1.33` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	29 mai 2025
Support supplémentaire pour le pilote Amazon FSx CSI	Vous pouvez désormais utiliser la console AWS de gestion, la AWS CLI et l'API pour gérer le pilote Amazon FSx CSI.	23 mai 2025
Modifiez les grattoirs Prometheus dans la console	Vous pouvez désormais modifier Amazon Managed Service for Prometheus scrapers dans la console Amazon EKS.	22 mai 2025

Mises à jour des politiques gérées	`AmazonEKSDashboardServiceRolePolicy` Politique ajoutée.	21 mai 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif de Kubernetes dans ces versions de plateforme.	16 mai 2025
Nouvelles pages pour Amazon FSx pour les performances de Lustre	Ajout de nouvelles rubriques contenant des informations détaillées sur l'optimisation des performances FSx d'Amazon pour Lustre.	2 mai 2025
Mise à jour du mode automatique d'Amazon EKS vers `NodeClass`	Le `NodeClass` modèle pour les nœuds en mode automatique a ajouté une configuration pour les proxys réseau de transfert. Cela ajoute la clé facultative `advancedNetworking` pour configurer votre proxy HTTPS.	30 avril 2025
Bottlerocket pour nœuds hybrides	Bottlerocket est désormais disponible pour les nœuds hybrides EKS.	29 avril 2025

Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif de Kubernetes dans ces versions de plateforme.	29 avril 2025
Nouvelles pages de concepts pour les réseaux hybrides	Des pages ont été ajoutées pour les concepts des nœuds hybrides EKS. Ils couvrent en détail les réseaux sur site et dans le cloud à l'aide de diagrammes.	18 avril 2025
AWS mises à jour des politiques gérées	Des autorisations ont été ajoutées `AmazonEKSClusterPolicy` pour autoriser Amazon EKS à accéder aux interfaces réseau élastiques créées par le VPC CNI. Cela est nécessaire pour qu'EKS puisse nettoyer les interfaces réseau élastiques qui restent en place en cas de fermeture inattendue du VPC CNI.	16 avril 2025
AWS mises à jour des politiques gérées	Des autorisations ont été ajoutées `AmazonEKSServiceRolePolicy` pour permettre aux AI/ML clients d'EKS d'ajouter des règles de sortie au groupe de sécurité par défaut du cluster EKS.	14 avril 2025

État des nœuds pour les nœuds hybrides EKS	Vous pouvez l'utiliser `eks-node-monitoring-agent` sur des nœuds hybrides, à partir de la version`1.2.0-eksbuild.1` . Exécutez-le `eks-node-monitoring-agent` en tant que module complémentaire Amazon EKS pour détecter et signaler les problèmes de santé.	31 mars 2025
Nœuds hybrides EKS pour les clusters existants	Vous pouvez désormais ajouter, modifier ou supprimer la configuration des nœuds hybrides des clusters existants . Auparavant, vous ne pouviez ajouter la configuration des nœuds hybrides aux nouveaux clusters que lorsque vous les créiez. Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et périphérique en tant que nœuds dans des clusters Amazon EKS. AWS gère le plan de contrôle Kubernetes AWS hébergé du cluster Amazon EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements locaux ou périphériques.	31 mars 2025

Annulation : évitez les mises à niveau accidentelles grâce à des informations sur les clusters	Amazon EKS a temporairement annulé une fonctionnalité qui vous obligeait à utiliser un `--force` indicateur pour mettre à niveau votre cluster en cas de problèmes d'analyse du cluster. Pour plus d'informations, voir Annulation temporaire de l'application des informations de mise à niveau lors de la mise à jour de la version du cluster sur. GitHub	28 mars 2025
Bottlerocket FIPS AMIs	Les Bottlerocket FIPS AMIs sont désormais disponibles dans les groupes de nœuds gérés standard.	27 mars 2025
AWS mises à jour des politiques gérées	Des autorisations ont été ajoutées `AmazonEKS ServiceRolePolicy` pour permettre à Amazon EKS de mettre fin aux EC2 instances créées par le mode automatique. Des autorisations ont été ajoutées `AmazonEKS ServiceRolePolicy` pour permettre à Amazon EKS de mettre fin aux EC2 instances créées par le mode automatique.	28 février 2025

Stratégies de mise à jour pour les groupes de nœuds gérés	Vous pouvez désormais utiliser des stratégies de mise à jour pour configurer le processus de mise à jour des versions pour les groupes de nœuds gérés. Cela introduit la stratégie de mise à jour minimale permettant de mettre fin aux nœuds avant d'en créer de nouveaux, ce qui est utile dans les environne ments à capacité limitée. La stratégie de mise à jour par défaut continue le comportem ent existant.	27 janvier 2025
Version de Kubernetes `1.32`	Ajout de la prise en `1.32` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	23 janvier 2025
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les régions Asie-Pacifique (Thaïlande) `ap-southeast-7` () et Mexique (centre) `mx-centra l-1` (AWS). Le mode automatique EKS et les points de terminaison VPC pour l'API EKS ne sont disponibles dans aucune des deux régions.	14 janvier 2025

AWS mises à jour des politiques gérées	Ajout de plusieurs autorisat ions pour permettre `AmazonEBSCSIDriver Policy` au pilote Amazon EBS CSI de restaurer tous les instantanés, d'activer la restauration rapide des instantanés (FSR) sur les volumes EBS et de modifier les balises sur les volumes.	13 janvier 2025
AWS mises à jour des politiques gérées	Autorisations ajoutées à`AmazonEKSLoadBalan cingPolicy` .	26 décembre 2024
Informations actualisées sur le cluster	Amazon EKS Upgrade Insights vous avertira désormais en cas d'autres problèmes liés à l'état du cluster et à la compatibilité des versions. Il peut détecter les problèmes entre les différent s composants Kubernetes et Amazon EKS`kubelet`, tels que, `kube-proxy` et les modules complémentaires Amazon EKS.	20 décembre 2024

Agent de surveillance des nœuds et réparation automatique	Vous pouvez utiliser le nouveau `eks-node-monitoring-agent` module complémentaire Amazon EKS pour détecter et signaler les problèmes de santé. Vous pouvez également activer la réparation automatique des nœuds pour remplacer automatiquement les nœuds lorsque des problèmes sont détectés.	16 décembre 2024
Nœuds hybrides Amazon EKS	Vous pouvez désormais exécuter un nœud sur site connecté à des clusters Amazon EKS. Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et périphérique en tant que nœuds dans des clusters Amazon EKS. AWS gère le plan de contrôle Kubernetes AWS hébergé du cluster Amazon EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements locaux ou périphériques.	1er décembre 2024

Mode automatique d'Amazon EKS	Le mode automatique Amazon EKS automatise entièrement la gestion de l'infrastructure de clusters Kubernetes pour le calcul, le stockage et la mise en réseau. AWS Il simplifie la gestion de Kubernetes en provisionnant automatiq uement l'infrastructure, en sélectionnant les instances de calcul optimales, en dimensionnant les ressource s de manière dynamique , en optimisant continuel lement les coûts, en appliquan t des correctifs aux systèmes d'exploitation et en intégrant les services de sécurité. AWS	1er décembre 2024
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorat ions de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes `1.31.2`, `1.30.6`, `1.29.10` et `1.28.15`.	22 novembre 2024
AWS mises à jour des politiques gérées	Mis à jour `AWSServic eRoleForAmazonEKSN odegroup` pour assurer la compatibilité avec les régions de Chine.	22 novembre 2024

La version 1.30 de Kubernetes est désormais disponible pour les clusters locaux sur Outposts AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un AWS Outposts à l'aide de Kubernetes version 1.30.	21 novembre 2024
AWS mises à jour des politiques gérées	Politique AWS gérée mise à jour par EKS`AmazonEKSLocalOutpostClusterPolicy` . `ec2:DescribeAvailabilityZones` Autorisation ajoutée afin que le AWS Cloud Controller Manager du plan de contrôle du cluster puisse identifier la zone de disponibilité dans laquelle se trouve chaque nœud.	21 novembre 2024
Bottlerocket utilisant la norme FIPS AMIs 140-3	AMIs Il existe des modèles Bottlerocket préconfigurés pour utiliser des modules cryptographiques validés par la norme FIPS 140-3. Cela inclut le module cryptographique de l'API Kernel Crypto d'Amazon Linux 2023 et le module cryptographique AWS-LC.	20 novembre 2024

AWS mises à jour des politiques gérées	`AWSServiceRoleForAmazonEKSNodegroup` Politique mise à jour `ec2:RebootInstances` pour autoriser les instances créées par des groupes de nœuds gérés par Amazon EKS. Limite les `ec2:CreateTags` autorisations pour les EC2 ressources Amazon.	20 novembre 2024
Tableau de bord d'observabilité	Le tableau de bord d'observabilité vous permet de détecter, de résoudre et de résoudre rapidement les problèmes . De nouvelles CloudWatch métriques vendues sont également disponibles dans l'espace de `AWS/EKS` noms.	18 novembre 2024
AWS mises à jour des politiques gérées	Politique AWS gérée mise à jour par EKS`AmazonEKSServiceRolePolicy` . Ajout d'autorisations pour les politiques d'accès EKS, la gestion de l'équilibreur de charge et le nettoyage automatique des ressources du cluster.	16 novembre 2024

Création de nouveaux rôles dans la console pour les modules complémentaires compatibles avec EKS Pod Identities	Lorsque vous utilisez la console pour créer ou mettre à jour des modules complémentaires compatibles avec EKS Pod Identities, de nouvelles étapes vous permettent de générer automatiquement des rôles IAM avec le nom, la politique de rôle et la politique de confiance appropriés pour le module complémentaire.	15 novembre 2024
Groupes de nœuds gérés dans les Zones AWS Locales	Les groupes de nœuds gérés peuvent désormais être créés dans les Zones AWS Locales.	15 novembre 2024
De nouvelles métriques sont disponibles	De nouvelles métriques sont disponibles dans le groupe d'APImetrics.eks.amazonaws.com .	11 novembre 2024
AWS mises à jour des politiques gérées	Politique AWS gérée mise à jour par EKSAmazonEKSComputePolicy . Autorisations de ressources mises à jour pour l'iam:AddRoleToInstanceProfile action.	7 novembre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : AmazonEKSComputePolicy	1er novembre 2024

AWS mises à jour des politiques gérées	Autorisations ajoutées à`AmazonEKSClusterPolicy` . Ajout `ec2:DescribeInstanceTopology` d'une autorisation permettant à Amazon EKS de joindre des informations topologiques au nœud sous forme d'étiquettes.	1er novembre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSBlockStoragePolicy`	30 octobre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSLoadBalancingPolicy`	30 octobre 2024
AWS mises à jour des politiques gérées	Des `cloudwatch:PutMetricData` autorisations ont été ajoutées `AmazonEKSServiceRolePolicy` pour permettre à Amazon EKS de publier des métriques sur Amazon CloudWatch.	29 octobre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSNetworkingPolicy`	28 octobre 2024

Points de terminaison à double pile pour les nouveaux clusters IPv6	Connectez-vous à de nouveaux IPv6 clusters avec un eks-cluster. *region*.api.aws point de terminaison à double pile. Ce point de terminaison est renvoyé lorsque vous décrivez ces clusters. kubectlet d'autres clients d'API Kubernetes installés dans IPv4 des environnements à double pile ou à double pile peuvent résoudre ces points de terminaison et s'y connecter pour des clusters publics ou privés. IPv6	21 octobre 2024
AWS mises à jour des politiques gérées	Autorisations ajoutées autoscaling:ResumeProcesses et associées pour les régions de Chine AWSServiceRoleForAmazonEKSNodegroup afin d'intégrer Amazon Application Recovery Controller for EKS. autoscaling:SuspendProcesses Aucun changement dans les autres régions.	21 octobre 2024
AL2023 accéléré AMIs	Vous pouvez désormais utiliser des instances accélérées NVIDIA et des instances AWS Neuron AMIs basées sur AL2 023.	11 octobre 2024

Nouveau format de source	Nous sommes passés à un nouveau format source avec quelques modifications de mise en page. Nous sommes en train de résoudre certains problèmes de formatage mineurs temporaires.	10 octobre 2024
AWS mises à jour des politiques gérées	Autorisations ajoutées à `AmazonEKSServicePolicy` et `AmazonEKSServiceRolePolicy` . Autorisations de balises ajoutées `ec2:GetSecurityGroupsForVpc` et associées pour permettre à EKS de lire les informations du groupe de sécurité et de mettre à jour les balises associées.	10 octobre 2024
AWS mises à jour des politiques gérées - Nouvelle politique	EKS a ajouté une nouvelle politique AWS gérée.	3 octobre 2024
Version de Kubernetes `1.31`	Ajout de la prise en `1.31` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	24 septembre 2024
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	21 août 2024

La version 1.29 de Kubernetes est désormais disponible pour les clusters locaux sur Outposts AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un AWS Outposts à l'aide de Kubernetes version 1.29.	20 août 2024
EKS Pod Identity aux États-Unis AWS GovCloud	Amazon EKS Pod Identities associe un rôle IAM à un compte de service Kubernetes. Grâce à cette fonction, vous n'avez plus besoin de fournir des autorisations étendues au rôle IAM du nœud. De cette façon, les pods de ce nœud peuvent appeler AWS APIs. Contrairement aux rôles IAM pour les comptes de service, les identités EKS Pod sont entièrement intégrées à EKS ; vous n'avez pas besoin d'un fournisseur d'identité OIDC.	14 août 2024
Mises à jour du contenu basées sur des scénarios	Nous avons renommé et mis à jour les rubriques afin qu'elles soient davantage axées sur des scénarios tout au long du guide.	9 août 2024
Points de terminaison d'interface VPC à double pile pour Amazon EKS	Vous pouvez désormais créer des points de terminaison d'interface VPC à double pile pour Amazon EKS `IPv4` avec à la `IPv6` fois des adresses IP et des noms DNS.	7 août 2024

Nouveaux points de terminaison à double pile pour Amazon EKS APIs avec adresses `IPv6`	L'API EKS pour la création et la gestion des clusters, et l'émetteur OIDC URLs pour les clusters disposent de nouveaux points de terminaison à double pile. Le nouveau nom DNS de l'API Amazon EKS est « `eks.`*`region`*`.api.aws` qui renvoie aux `IPv4` adresses et aux `IPv6` adresses ». Les nouveaux clusters ont une nouvelle URL d'émetteur OIDC à double pile (). `oidc-eks.`*`region`*`.api.aws`	1er août 2024
Blocs de capacité pour les groupes de nœuds gérés	Vous pouvez désormais utiliser des blocs de capacité pour les groupes de nœuds gérés.	1 juillet 2024
La collecte de métriques Auto Scaling Group est activée par défaut	Les métriques de groupe Amazon EC2 Auto Scaling sont désormais activées par défaut pour les groupes de nœuds gérés par Amazon EKS, sans frais supplémentaires. Auparavant, vous deviez effectuer plusieurs étapes pour activer cette fonctionnalité.	28 juin 2024
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	27 juin 2024

Version de Kubernetes `1.26`	La version Kubernetes bénéficie désormais d'`1.26`un support étendu.	12 juin 2024
Améliorations apportées aux références d'informations de l'AMI	Nous avons apporté des améliorations aux référence s d'informations de l'AMI, en particulier pour Bottlerocket.	12 juin 2024
Version de Kubernetes `1.30`	Ajout de la prise en `1.30` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	23 mai 2024
Mise à l'échelle automatique du CoreDNS	L'autoscaler CoreDNS adaptera dynamiquement le nombre de répliques du déploiement CoreDNS dans un cluster EKS en fonction du nombre de nœuds et de cœurs de processeur. Cette fonctionnalité fonctionne pour `v1.9` CoreDNS et les versions les plus récentes de la plateforme EKS et versions `1.25` ultérieures.	14 mai 2024
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes`1.29.4`,, `1.28.9` et. `1.27.13`	14 mai 2024

CloudWatch Support de Container Insights pour Windows	Le module complémentaire Amazon CloudWatch Observability Operator autorise désormais également Container Insights sur les nœuds de travail Windows du cluster.	10 avril 2024
Concepts de Kubernetes	Ajout d'une nouvelle rubrique sur les concepts de Kubernetes.	5 avril 2024
Restructurer l'accès et le contenu IAM	Déplacez les pages existantes liées aux sujets relatifs à l'accès et à l'IAM, telles que la carte de configuration d'authentification, les entrées d'accès, l'ID du pod et l'IRSA dans une nouvelle section. Réviser le contenu de l'aperçu.	2 avril 2024
Support du système d'exploitation Bottlerocket pour le pilote Amazon S3 CSI	Le pilote CSI Mountpoint pour Amazon S3 est désormais compatible avec Bottlerocket.	13 mars 2024
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	4 mars 2024
Amazon Linux 2023	Amazon Linux 2023 (AL2023) est un nouveau système d'exploitation basé sur Linux conçu pour fournir un environnement sécurisé, stable et performant pour vos applications cloud.	29 février 2024

EKS Pod Identity et IRSA prennent en charge les sidecars dans Kubernetes `1.29`	Dans Kubernetes`1.29`, les conteneurs annexes sont disponibles dans les clusters Amazon EKS. Les conteneurs annexes sont pris en charge avec des rôles IAM pour les comptes de service ou EKS Pod Identity. Pour plus d'informations sur les sidecars, consultez la section Sidecar Containers dans la documentation de Kubernetes.	26 février 2024
Version de Kubernetes `1.29`	Ajout de la prise en `1.29` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	23 janvier 2024
Version complète : Support étendu d'Amazon EKS pour les versions de Kubernetes	La prise en charge étendue des versions de Kubernetes vous permet de conserver une version spécifique de Kubernetes pendant plus de 14 mois.	16 janvier 2024

Détection de l'état du cluster Amazon EKS dans le AWS cloud	Amazon EKS détecte les problèmes liés à vos clusters Amazon EKS et à l'infrast ructure des conditions requises pour le cluster en termes de santé du cluster. Vous pouvez consulter les problèmes liés à vos clusters EKS dans AWS Managemen t Console et dans `health` le cluster dans l'API EKS. Ces problèmes s'ajoutent aux problèmes détectés et affichés par la console. Auparavant, l'état de santé des clusters n'était disponible que pour les clusters locaux sur AWS Outposts.	28 décembre 2023
Informations sur les clusters	Vous pouvez désormais obtenir des recommandations sur votre cluster sur la base de vérifications récurrentes.	20 décembre 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Ouest du Canada (Calgary`ca-west-1`) ().	20 décembre 2023

Vous pouvez désormais accorder aux utilisateurs et rôles IAM l'accès à votre cluster à l'aide d'entrées d'accès	Avant l'introduction des entrées d'accès, vous avez accordé aux utilisateurs et rôles IAM l'accès à votre cluster en ajoutant des entrées à la `ConfigMap aws-auth`. Chaque cluster dispose désormais d'un mode d'accès, et vous pouvez passer à l'utilisation des entrées d'accès selon votre calendrier. Après avoir changé de mode, vous pouvez ajouter des utilisate urs en ajoutant des entrées d'accès dans la AWS CLI AWS CloudFormation, et le AWS SDKs.	18 décembre 2023
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorat ions de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes `1.28.4`, `1.27.8`, `1.26.11` et `1.25.16`.	12 décembre 2023
Pilote CSI Mountpoint pour Amazon S3	Vous pouvez désormais installer le pilote CSI Mountpoint pour Amazon S3 sur les clusters Amazon EKS.	27 novembre 2023

Activez les métriques Prometheus lors de la création d'un cluster	Dans le AWS Management Console, vous pouvez désormais activer les métriques Prometheus lors de la création d'un cluster. Vous pouvez également consulter les détails du grattoir Prometheus dans l'onglet Observabilité.	26 novembre 2023
Identités du pod Amazon EKS	Amazon EKS Pod Identities associe un rôle IAM à un compte de service Kubernetes. Grâce à cette fonction, vous n'avez plus besoin de fournir des autorisations étendues au rôle IAM du nœud. De cette façon, les pods de ce nœud peuvent appeler AWS APIs. Contrairement aux rôles IAM pour les comptes de service, les identités EKS Pod sont entièrement intégrées à EKS ; vous n'avez pas besoin d'un fournisseur d'identité OIDC.	26 novembre 2023
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	26 novembre 2023
Contrôleur d'instantané CSI	Vous pouvez désormais installer le contrôleur d'instantané CSI pour l'utiliser avec des pilotes CSI compatibles, tels que le pilote CSI Amazon EBS.	17 novembre 2023

Réécriture des rubriques de l'opérateur ADOT	La section relative à la prise en charge du module complémentaire Amazon EKS pour ADOT Operator était redondante avec le AWS Distro pour ce qui est de la documentation. OpenTelem etry Nous avons migré les informations essentielles restantes vers cette ressource afin de réduire les informations obsolètes et incohérentes.	14 novembre 2023
Support du module complémentaire CoreDNS EKS pour les métriques Prometheus	Les `v1.8.7-eksbuild.8` versions `v1.10.1-eksbuild.5` `v1.9.3-eksbuild.9` , et du module complémentaire EKS pour CoreDNS exposent le port sur lequel CoreDNS a publié les métriques dans le service. `kube-dns` Cela facilite l'inclusion des métriques CoreDNS dans vos systèmes de surveillance.	10 novembre 2023
Module complémentaire Amazon EKS CloudWatch Observability Operator	Ajout de la page Amazon EKS CloudWatch Observability Operator.	6 novembre 2023
Blocs de capacité pour les instances P5 autogérées aux USA Est (Ohio)	Aux USA Est (Ohio), vous pouvez désormais utiliser des blocs de capacité pour les instances P5 autogérées.	31 octobre 2023

Les clusters prennent en charge des sous-réseaux et des groupes de sécurité	Vous pouvez mettre à jour le cluster pour modifier les sous-réseaux et les groupes de sécurité qu'il utilise. Vous pouvez effectuer la AWS Management Console mise à jour à partir de la dernière version de la AWS CLI et de `eksctl` la version `v0.164.0-rc.0` ou ultérieure. AWS CloudFormation Vous pouvez avoir besoin de procéder ainsi pour fournir aux sous-réseaux davantage d'adresses IP disponibles afin de réussir la mise à niveau d'une version de cluster.	24 octobre 2023
Le rôle de cluster et le rôle de groupe de nœuds gérés prennent en charge les politiques AWS d'Identity and Access Management gérées par le client	Vous pouvez utiliser une politique IAM personnalisée sur le rôle de cluster, au lieu de la EKSClusterpolitique AWS gérée par Amazon Policy. Vous pouvez également utiliser une politique IAM personnalisée sur le rôle du nœud dans un groupe de nœuds gérés, au lieu de la politique EKSWorker NodePolicy AWS gérée par Amazon. Cela permet de créer une politique avec le moins de privilèges possible afin de répondre à des exigences strictes en matière de conformité.	23 octobre 2023

Correction du lien vers l'installation d'eksctl	Correction du lien d'installation d'eksctl après que la page ait été déplacée.	6 octobre 2023
Version préliminaire : Support étendu d'Amazon EKS pour les versions de Kubernetes	La prise en charge étendue des versions de Kubernetes vous permet de conserver une version spécifique de Kubernetes pendant plus de 14 mois.	4 octobre 2023
Supprimer les références à AWS l'intégration d'App Mesh	Les intégrations d'Amazon EKS avec AWS App Mesh restent réservées aux clients existants d'App Mesh.	29 septembre 2023
Version de Kubernetes `1.28`	Ajout de la prise en `1.28` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	26 septembre 2023
Les clusters existants prennent en charge l'application de la politique réseau de Kubernetes dans le plug-in Amazon VPC CNI pour Kubernetes.	Vous pouvez utiliser la politique réseau Kubernetes dans les clusters existants avec le plug-in Amazon VPC CNI pour Kubernetes, au lieu d'avoir besoin d'une solution tierce. Vous pouvez utiliser la politique réseau Kubernetes dans les clusters existants avec le plug-in Amazon VPC CNI pour Kubernetes, au lieu d'avoir besoin d'une solution tierce.	15 septembre 2023

Le module complémentaire CoreDNS Amazon EKS prend en charge la modification du PDB	Vous pouvez modifier le `PodDisruptionBudget` module complémentaire EKS pour CoreDNS dans les `v1.9.3-eksbuild.7` versions et ultérieures`v1.10.1-eksbuild.4` .	15 septembre 2023
Prise en charge d'Amazon EKS pour les sous-réseaux partagés	Nouvelles exigences et considérations relatives aux sous-réseaux partagéspour créer des clusters Amazon EKS dans des sous-réseaux partagés.	7 septembre 2023
Mises à jour de Qu'est-ce qu'Amazon EKS ?	Ajout de nouveaux cas d'utilisation courants et de nouvelles rubriques Architecture. Autres rubriques actualisées.	6 septembre 2023
Application de la politique réseau Kubernetes dans le plug-in Amazon VPC CNI pour Kubernetes	Vous pouvez utiliser la politique réseau Kubernetes avec le plug-in Amazon VPC CNI pour Kubernetes, au lieu d'avoir besoin d'une solution tierce. Vous pouvez utiliser la politique réseau Kubernetes avec le plug-in Amazon VPC CNI pour Kubernetes, au lieu d'avoir besoin d'une solution tierce.	29 août 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la région Israël (Tel Aviv) `il-central-1` (AWS).	1er août 2023

Stockage éphémère configurable pour Fargate	Vous pouvez augmenter la quantité totale de stockage éphémère pour chaque pod exécuté sur Amazon EKS Fargate.	31 juillet 2023
Prise en charge du module complémentaire pour le pilote CSI d'Amazon EFS	Vous pouvez désormais utiliser la AWS Management Console AWS CLI et l'API pour gérer le pilote Amazon EFS CSI.	26 juillet 2023
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	26 juillet 2023
Les mises à jour des versions 1.27, 1.26, 1.25 et 1.24 de Kubernetes sont désormais disponibles pour les clusters locaux sur Outposts AWS	Les mises à jour des versions 1.27.3, 1.26.6, 1.25.11 et 1.24.15 de Kubernetes sont désormais disponibles pour les clusters locaux sur Outposts AWS	20 juillet 2023
Prise en charge des préfixes IP pour les nœuds Windows	L'attribution de préfixes IP à vos nœuds peut vous permettre d'héberger un nombre nettement plus élevé de pods sur vos nœuds que lorsque vous attribuez des adresses IP secondaires individuelles à vos nœuds.	6 juillet 2023
Pilote FSx CSI Amazon pour OpenZFS	Vous pouvez désormais installer le pilote CSI Amazon FSx pour OpenZFS sur les clusters Amazon EKS.	30 juin 2023

Les pods des nœuds Linux situés dans des `IPv4` clusters peuvent désormais communiquer avec les `IPv6` points de terminaison.	Après avoir attribué une IPv6 adresse à votre nœud, l'`IPv4`adresse de vos pods est l'adresse réseau traduite en `IPv6` adresse du nœud sur lequel ils s'exécutent.	19 juin 2023
Groupes de nœuds gérés par Windows dans les régions AWS GovCloud (États-Unis)	Dans les régions AWS GovCloud (États-Unis), les groupes de nœuds gérés par Amazon EKS peuvent désormais exécuter des conteneurs Windows.	30 mai 2023
Version de Kubernetes `1.27`	Ajout de la prise en `1.27` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	24 mai 2023
Version de Kubernetes `1.26`	Ajout de la prise en `1.26` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	11 avril 2023
GMSA sans domaine	Vous pouvez désormais utiliser des GMSA sans domaine avec Windows Pods.	27 mars 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Melbourne`ap-southeast-4`) ().	10 mars 2023

Pilote CSI Amazon File Cache	Vous pouvez désormais installer le pilote CSI Amazon File Cache sur les clusters Amazon EKS.	3 mars 2023
La version 1.25 de Kubernetes est désormais disponible pour les clusters locaux sur Outposts AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un Outpost à l'aide des versions de Kubernetes —. `1.22` `1.25`	1er mars 2023
Version de Kubernetes `1.25`	Ajout de la prise en `1.25` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	22 février 2023
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	7 février 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les régions Asie-Pacifique (Hyderabad) (`ap-south-2`), Europe (Zurich) (`eu-central-2`) et Europe (Espagne) (`eu-south-2`). AWS	6 février 2023
Versions de Kubernetes `1.21` : `1.24` elles sont désormais disponibles pour les clusters locaux sur Outposts. AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un Outpost à l'aide des versions de Kubernetes —. `1.21` `1.24` Auparavant, seule la version `1.21` était disponible.	17 janvier 2023

Amazon EKS prend désormais en charge AWS PrivateLink	Vous pouvez utiliser an AWS PrivateLink pour créer une connexion privée entre votre VPC et Amazon EKS.	16 décembre 2022
Prise en charge par Windows des groupes de nœuds gérés	Vous pouvez désormais utiliser Windows pour les groupes de nœuds gérés Amazon EKS.	15 décembre 2022
Les modules complémentaires Amazon EKS proposés par des éditeurs de logiciels indépendants sont désormais disponibles AWS sur le Marketplace	Vous pouvez désormais parcourir les modules complémentaires Amazon EKS proposés par des fournisseurs de logiciels indépendants et vous y abonner via le AWS Marketplace.	28 novembre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	17 novembre 2022
Version de Kubernetes `1.24`	Ajout de la prise en `1.24` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	15 novembre 2022
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Moyen-Orient (Émirats arabes unisme`-central-1`) ().	3 novembre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	24 octobre 2022

AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	20 octobre 2022
Les clusters locaux sur AWS Outposts sont désormais disponibles	Vous pouvez désormais créer un cluster local Amazon EKS sur un Outpost.	19 septembre 2022
Quotas basés sur les vCPU Fargate	Fargate est en train de passer des quotas basés sur les pods aux quotas basés sur les vCPU.	8 septembre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	31 août 2022
Suivi des coûts	Amazon EKS prend désormais en charge Kubecost, ce qui vous permet de surveiller les coûts ventilés par ressources Kubernetes, notamment les pods, les nœuds, les espaces de noms et les étiquettes.	24 août 2022
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	24 août 2022
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	23 août 2022
Balisage des ressources pour la facturation	Ajout de la prise en charge de balises de répartition des coûts généré par `aws:eks:cluster-name` pour tous les clusters.	16 août 2022

Caractères génériques du profil de Fargate	Ajout de la prise en charge des caractères génériques du profil Fargate dans les critères de sélection pour les espaces de noms, les clés d'étiquette et les valeurs d'étiquette.	16 août 2022
Version de Kubernetes `1.23`	Ajout de la prise en `1.23` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	11 août 2022
Consultez les ressources Kubernetes dans le AWS Management Console	Vous pouvez désormais consulter les informations relatives aux ressources Kubernetes déployées sur votre cluster à l'aide du. AWS Management Console	3 mai 2022
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Jakartaap-`southeast-3`) ().	2 mai 2022
Prise en charge de la page Observability et du module complémentaire ADOT	Ajout d'une page d'observa bilité et d' AWS une distribut ion pour OpenTelemetry (ADOT).	21 avril 2022
Version de Kubernetes `1.22`	Ajout de la prise en `1.22` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	4 avril 2022

AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	4 avril 2022
Ajout de détails sur les correctifs de Fargate Pod	Lors de la mise à niveau de Fargate Pods, Amazon EKS essaie d'abord d'expulser les Pods en fonction de votre budget d'interruption de service. Vous pouvez créer des règles d'événement pour réagir aux expulsions ratées avant que les pods ne soient supprimés.	1er avril 2022
Version complète : prise en charge du module complémentaire pour le pilote CSI d'Amazon EBS	Vous pouvez désormais utiliser la AWS Management Console AWS CLI et l'API pour gérer le pilote Amazon EBS CSI.	31 mars 2022
AWS Mise à jour du contenu d'Outposts	Instructions pour déployer un cluster Amazon EKS sur AWS Outposts.	22 mars 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	21 mars 2022
Prise en charge de Windows `containerd`	Vous pouvez désormais sélectionner l'exécution `containerd` pour les nœuds Windows.	14 mars 2022
Ajout de considérations Amazon EKS Connector à la documentation de sécurité	Décrit le modèle de responsabilité partagée en ce qui concerne les clusters connectés.	25 février 2022

Attribuez des `IPv6` adresses à vos pods et services	Vous pouvez désormais créer un cluster `1.21` ou une version ultérieure qui attribue des `IPv6` adresses à vos pods et services.	6 janvier 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	13 décembre 2021
Version préliminaire : prise en charge du module complémentaire pour le pilote CSI d'Amazon EBS	Vous pouvez désormais prévisualiser à l' AWS Management Console aide de la AWS CLI et de l'API pour gérer le pilote Amazon EBS CSI.	9 décembre 2021
Prise en charge de Karpenter Autoscaler	Vous pouvez désormais utiliser le projet open source Karpenter pour mettre à l'échelle automatique vos nœuds.	29 novembre 2021
Prise en charge du filtre Kubernetes Fluent Bit dans la journalisation Fargate	Vous pouvez désormais utiliser le filtre Kubernetes Fluent Bit avec la journalisation Fargate.	10 novembre 2021
Prise en charge de Windows disponible dans le plan de contrôle	La prise en charge de Windows est désormais disponible dans votre plan de contrôle. Vous n'avez plus besoin de l'activer dans votre plan de données.	9 novembre 2021

Bottlerocket ajouté en tant que type d'AMI pour les groupes de nœuds gérés	Auparavant, Bottlerocket était uniquement disponible en tant qu'option de nœud autogéré. Il peut désormais être configuré en tant que groupe de nœuds gérés, ce qui réduit les efforts nécessaires pour répondre aux exigences de conformité des nœuds.	28 octobre 2021
DL1 assistance au conducteur	Amazon Linux personnalisé prend AMIs désormais en charge les charges de travail d'apprentissage profond pour Amazon Linux 2. Cette activation permet une configuration de base générique sur site ou dans le cloud.	25 octobre 2021
VT1 support vidéo	Amazon Linux personnalisé AMIs est désormais compatible VT1 avec certaines distributions. Cette mise en œuvre annonce les dispositifs Xilinx U30 sur votre cluster Amazon EKS.	13 septembre 2021
Amazon EKS Connector est désormais disponible	Vous pouvez utiliser le connecteur Amazon EKS pour enregistrer et connecter tout cluster Kubernetes conforme à la console Amazon EKS AWS et le visualiser dans la console Amazon EKS.	8 septembre 2021

Amazon EKS Anywhere est désormais disponible	Amazon EKS Anywhere est une nouvelle option de déploiement pour Amazon EKS que vous pouvez utiliser pour créer et exploiter des clusters Kubernetes sur site.	8 septembre 2021
Pilote Amazon FSx pour NetApp ONTAP CSI	Ajout d'une rubrique qui résume le pilote Amazon FSx for NetApp ONTAP CSI et fournit des liens vers d'autres références.	2 septembre 2021
Les groupes de nœuds gérés calculent désormais automatiquement le nombre maximal de pods recommandé par Amazon EKS pour les nœuds	Les groupes de nœuds gérés calculent désormais automatiquement le nombre maximum de pods Amazon EKS pour les nœuds que vous déployez sans modèle de lancement ou avec un modèle de lancement dans lequel vous n'avez pas spécifié d'ID AMI.	30 août 2021
Supprimer la gestion Amazon EKS des paramètres du module complémentaire sans supprimer le logiciel complémentaire Amazon EKS	Vous pouvez désormais supprimer un module complémentaire Amazon EKS sans supprimer le logiciel complémentaire de votre cluster.	20 août 2021
Créer des pods multi-résidents à l'aide de Multus	Vous pouvez maintenant ajouter plusieurs interfaces réseau à un pod à l'aide de Multus.	2 août 2021

Ajoutez d'autres adresses IP à vos EC2 nœuds Amazon Linux	Vous pouvez désormais ajouter beaucoup plus d'adresses IP à vos EC2 nœuds Amazon Linux. Cela signifie que vous pouvez exécuter une plus grande densité de pods sur chaque nœud. Vous pouvez désormais ajouter beaucoup plus d'adresses IP à vos EC2 nœuds Amazon Linux. Cela signifie que vous pouvez exécuter une plus grande densité de pods sur chaque nœud.	27 Juillet 2021
Amorçage d'exécution `containerd`	L'Amazon Machine Image (AMI) accélérée optimisée pour Amazon EKS contient désormais un indicateur de démarrage que vous pouvez utiliser pour activer l'`containerd` exécution dans Amazon EKS Optimized et AMIs Bottlerocket. Cet indicateur est disponible dans toutes les versions de Kubernetes prises en charge de l'AMI.	19 juillet 2021
Version de Kubernetes `1.21`	Ajout du support des versions de Kubernetes. `1.21`	19 juillet 2021

Ajout d'une rubrique de politiques gérées	Liste de toutes les politiques gérées par Amazon EKS IAM et des modifications qui leur ont été apportées depuis le 17 juin 2021.	17 juin 2021
Utiliser des groupes de sécurité pour Pods with Fargate	Vous pouvez désormais utiliser des groupes de sécurité pour les pods avec Fargate, en plus de les utiliser avec les nœuds Amazon. EC2	1er juin 2021
Ajout de CoreDNS et de modules complémentaires `kube-proxy` Amazon EKS	Amazon EKS peut désormais vous aider à gérer les CoreDNS et les modules complémentaires Amazon EKS `kube-proxy` pour votre cluster.	19 mai 2021
Version de Kubernetes `1.20`	Ajout de la prise en `1.20` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	18 mai 2021
AWS Lancement du Load Balancer Controller `2.2.0`	Vous pouvez désormais utiliser le AWS Load Balancer Controller pour créer des Elastic Load Balancers à l'aide d'instances ou de cibles IP.	14 mai 2021
Taints de noeuds pour les groupes de nœuds gérés	Amazon EKS prend désormais en charge l'ajout de taints de notes aux groupes de nœuds gérés.	11 mai 2021

Chiffrement des secrets pour les clusters existants	Amazon EKS prend désormais en charge l'ajout du chiffrement des secrets aux clusters existants.	26 février 2021
Version de Kubernetes `1.19`	Ajout de la prise en `1.19` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	16 février 2021
Amazon EKS prend désormais en charge les fournisseurs d'identité OpenID Connect (OIDC) comme méthode d'authentification des utilisateurs d'un cluster de version `1.16` ou ultérieure.	Les fournisseurs d'identité OIDC peuvent être utilisés avec ou comme alternative à AWS Identity and Access Management (IAM).	12 février 2021
Afficher les ressources du nœud et de la charge de travail dans AWS Management Console	Vous pouvez désormais consulter les détails de vos nœuds gérés, autogérés et Fargate ainsi que de vos charges de travail Kubernetes déployées dans le. AWS Management Console	1er décembre 2020
Déployer des types d'instances Spot dans un groupe de nœuds gérés	Vous pouvez désormais déployer plusieurs types d'instances Spot ou à la demande dans un groupe de nœuds gérés.	1er décembre 2020

Amazon EKS peut désormais gérer des modules complémentaires spécifiques pour votre cluster	Vous pouvez gérer vous-même les modules complémentaires ou autoriser Amazon EKS à contrôler le lancement et la version d'un module complémentaire via l'API Amazon EKS.	1er décembre 2020
Partager un ALB sur plusieurs entrées	Vous pouvez désormais partager un AWS Application Load Balancer (ALB) entre plusieurs entrées Kubernetes. Auparavant, vous deviez déployer un ALB distinct pour chaque entrée.	23 octobre 2020
Prise en charge des cibles IP NLB	Vous pouvez désormais déployer un Network Load Balancer avec des cibles IP. Cela signifie que vous pouvez utiliser un NLB pour équilibrer la charge du trafic réseau vers les Fargate Pods et directement vers les Pods exécutés sur les nœuds Amazon. EC2	23 octobre 2020
Version de Kubernetes `1.18`	Ajout de la prise en `1.18` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	13 octobre 2020
Spécifiez un bloc CIDR personnalisé pour l'attribution de l'adresse IP du service Kubernetes.	Vous pouvez désormais spécifier un bloc CIDR personnalisé à partir duquel Kubernetes attribue les adresses IP de service.	29 septembre 2020

Attribuer des groupes de sécurité à des pods individuels	Vous pouvez désormais associer différents groupes de sécurité à certains des pods individuels qui s'exécutent sur de nombreux types d' EC2 instances Amazon.	9 septembre 2020
Déployer Bottlerocket sur vos nœuds	Vous pouvez désormais déployer des nœuds qui exécutent Bottlerocket.	31 août 2020
La possibilité de lancer des nœuds Arm est généralement disponible	Vous pouvez maintenant lancer des nœuds Arm dans des groupes de nœuds gérés et autogérés.	17 août 2020
Modèles de lancement de groupes de nœuds gérés et AMI personnalisée	Vous pouvez désormais déployer un groupe de nœuds gérés qui utilise un modèle de EC2 lancement Amazon. Le modèle de lancement peut spécifier une AMI personnal isée, si vous le souhaitez.	17 août 2020
Support EFS pour AWS Fargate	Vous pouvez désormais utiliser Amazon EFS avec AWS Fargate.	17 août 2020

Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut le support UDP pour les services de ce type `LoadBalancer` lors de l'utilisation d'équilibreurs de charge réseau avec la version Kubernetes ou ultérieure. `1.15` Pour plus d'informations, consultez le problème Allow UDP for AWS Network Load Balancer sur. GitHub	12 août 2020
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les AWS régions Afrique (Le Capa`f-south-1`) () et Europe (Milane`u-south-1`) ().	6 août 2020
Métriques d'utilisation de Fargate	AWS Fargate CloudWatch fournit des statistiques d'utilisation qui fournissent une visibilité sur l'utilisation des ressources Fargate On-Demand par votre compte.	3 août 2020
Version de Kubernetes `1.17`	Ajout de la prise en `1.17` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	10 juillet 2020

Créer et gérer des ressources App Mesh depuis Kubernetes avec le contrôleur App Mesh for Kubernetes	Vous pouvez créer et gérer des ressources App Mesh depuis Kubernetes. Le contrôleur injecte également automatiquement le proxy Envoy et les conteneurs d'initialisation dans les pods que vous déployez.	18 juin 2020
Amazon EKS prend désormais en charge les nœuds Amazon EC2 Inf1	Vous pouvez ajouter des nœuds Amazon EC2 Inf1 à votre cluster.	4 juin 2020
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les AWS GovCloud régions (USA Est) (`us-gov-east-1`) et AWS GovCloud (USA Ouest) (`us-gov-west-1`). AWS	13 mai 2020
Kubernetes n'`1.12`est plus pris en charge sur Amazon EKS	La version Kubernetes n'`1.12`est plus prise en charge sur Amazon EKS. Veuillez mettre à jour les clusters `1.12` vers la version `1.13` ou une version ultérieure pour éviter toute interruption de service.	12 mai 2020
Version de Kubernetes `1.16`	Ajout de la prise en `1.16` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	30 avril 2020

Ajout du rôle lié au service AWSServiceRoleForAmazonEKS	Le rôle lié au service AWSServiceRoleForAmazonEKS a été ajouté.	16 avril 2020
Version de Kubernetes `1.15`	Ajout de la prise en `1.15` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	10 mars 2020
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les AWS régions de Pékin (`cn-north-1`) et de Ningxia (`cn-northwest-1`).	26 février 2020
FSx pour le pilote Lustre CSI	Ajout d'une rubrique pour l'installation du pilote CSI FSx for Lustre sur les clusters Kubernetes `1.14` Amazon EKS.	23 décembre 2019
Restreindre l'accès réseau au point de terminaison d'accès public d'un cluster	Avec cette mise à jour, vous pouvez utiliser Amazon EKS pour restreindre les plages CIDR qui peuvent communiquer avec le point de terminaison d'accès public du serveur API Kubernetes.	20 décembre 2019
Résoudre l'adresse du point de terminaison de l'accès privé pour un cluster depuis l'extérieur d'un VPC	Avec cette mise à jour, vous pouvez utiliser Amazon EKS pour résoudre le point de terminaison d'accès privé du serveur API Kubernetes depuis l'extérieur d'un VPC.	13 décembre 2019

(Bêta) Nœuds d' EC2 instance Amazon EC2 A1 Amazon	Lancez les nœuds d' EC2 instance Amazon EC2 A1 enregistrés auprès de votre cluster Amazon EKS.	4 décembre 2019
Création d'un cluster sur AWS Outposts	Amazon EKS prend désormais en charge la création de clusters sur AWS Outposts.	3 décembre 2019
AWS Fargate sur Amazon EKS	Les clusters Amazon EKS Kubernetes prennent désormais en charge l'exécuti on de Pods sur Fargate.	3 décembre 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Canada (Centre`ca-central-1`) ().	21 novembre 2019
Groupes de nœuds gérés	Les groupes de nœuds gérés par Amazon EKS automatis ent le provisionnement et la gestion du cycle de vie des nœuds (EC2 instances Amazon) pour les clusters Amazon EKS Kubernetes.	18 novembre 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelles versions de plateforme pour résoudre CVE-2019-11253.	6 novembre 2019

Kubernetes n'`1.11`est plus pris en charge sur Amazon EKS	La version Kubernetes n'`1.11`est plus prise en charge sur Amazon EKS. Veuillez mettre à jour les clusters `1.11` vers la version `1.12` ou ultérieure pour éviter toute interruption de service.	4 novembre 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Amérique du Sud (São Paulo`sa-east-1`) ().	16 octobre 2019
Prise en charge de Windows	Les clusters Amazon EKS exécutant la version Kubernetes prennent `1.14` désormais en charge les charges de travail Windows.	7 octobre 2019
Autoscaling	Ajout d'un chapitre pour couvrir certains des différents types de scalabilité automatique Kubernetes pris en charge dans les clusters Amazon EKS.	30 septembre 2019
Mise à jour du tableau de bord Kubernetes	Rubrique mise à jour concernant l'installation du tableau de bord Kubernetes sur les clusters Amazon EKS pour utiliser la version bêta. `2.0`	28 septembre 2019

Pilote CSI Amazon EFS	Ajout d'une rubrique pour l'installation du pilote Amazon EFS CSI sur les clusters `1.14` Amazon EKS de Kubernetes.	19 septembre 2019
Paramètre Amazon EC2 Systems Manager pour l'ID d'AMI optimisé pour Amazon EKS	Ajout d'une rubrique permettant de récupérer l'ID d'AMI optimisé pour Amazon EKS à l'aide d'un paramètre Amazon EC2 Systems Manager. Ce paramètre vous évite d'avoir à rechercher l'AMI IDs.	18 septembre 2019
Étiquetage des ressources Amazon EKS	Vous pouvez gérer l'identification de vos clusters Amazon EKS.	16 septembre 2019
Pilote CSI Amazon EBS	Ajout d'une rubrique pour l'installation du pilote Amazon EBS CSI sur les clusters Amazon EKS de Kubernetes. `1.14`	9 septembre 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour `CVE-2019-9512` et `CVE-2019-9514`	Amazon EKS a mis à jour l'AMI optimisée pour Amazon EKS pour résoudre CVE-2019-9512 et CVE-2019-9514.	6 septembre 2019
Annonce de la dépréciation de `1.11` Kubernetes dans Amazon EKS	Amazon EKS a interrompu le support de la version Kubernetes le 4 `1.11` novembre 2019.	4 septembre 2019
Version de Kubernetes `1.14`	Ajout de la prise en `1.14` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	3 septembre 2019

Rôles IAM pour les comptes de service	Avec les rôles IAM pour les comptes de service sur les clusters Amazon EKS, vous pouvez associer un rôle IAM à un compte de service Kubernetes. Grâce à cette fonction, vous n'avez plus besoin de fournir des autorisations étendues au rôle IAM du nœud. De cette façon, les pods de ce nœud peuvent appeler AWS APIs.	3 septembre 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la région Moyen-Orient (Bahreïn) `me-south-1` (AWS).	29 août 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelles versions de plateforme pour résoudre CVE-2019-9512 et CVE-2019-9514.	28 août 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelles versions de plateforme pour résoudre CVE-2019-11247 et CVE-2019-11249.	5 août 2019
Extension de région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Hong Kong`ap-east-1`) ().	31 juillet 2019

Kubernetes `1.10` n'est plus pris en charge sur Amazon EKS	La version Kubernetes n'`1.10`est plus prise en charge sur Amazon EKS. Mettez à jour tous les clusters `1.10` vers la version `1.11` ou une version ultérieure pour éviter toute interruption de service.	30 juillet 2019
Ajout d'une rubrique sur le contrôleur d'entrée ALB	Le contrôleur d'entrée AWS ALB pour Kubernetes est un contrôleur qui provoque la création d'un ALB lors de la création de ressources d'entrée.	11 juillet 2019
Nouvelle AMI optimisée pour Amazon EKS	Supprimer le `kubectl` binaire inutile de AMIs.	3 juillet 2019
Version de Kubernetes `1.13`	Ajout de la prise en `1.13` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	18 juin 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour `AWS-2019-005`	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS pour corriger les vulnérabi lités décrites dans link:secu rity/security-bulletins/AWS -2019-005/ [-2019-005, type="marketing "].AWS	17 juin 2019
Annonce de l'arrêt du support de Kubernetes `1.10` dans Amazon EKS	Amazon EKS a cessé de prendre en charge la version Kubernetes le 22 `1.10` juillet 2019.	21 mai 2019

Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateform e pour Kubernetes `1.11` et les `1.10` clusters afin de prendre en charge les noms DNS personnalisés dans le `kubelet` certificat et d'amélior er les performances. `etcd`	21 mai 2019
Démarrer avec `eksctl`	Ce guide de démarrage explique comment installer toutes les ressources nécessaires pour démarrer avec Amazon EKS en utilisant `eksctl`. Il s'agit d'un utilitaire de ligne de commande simple pour créer et gérer les clusters Kubernetes sur Amazon EKS.	10 mai 2019

AWS`get-token` commande CLI	La `aws eks get-token` commande a été ajoutée à la AWS CLI. Il n'est plus nécessaire d'installer l'authentificateur AWS IAM pour Kubernetes pour créer des jetons de sécurité client pour les communications avec le serveur d'API du cluster. Mettez à niveau votre installation AWS CLI vers la dernière version pour utiliser cette nouvelle fonctionnalité. Pour plus d'informations, consultez la section Installation de l'interface de ligne de AWS commande dans le guide de l'utilisateur de l'interface de ligne de AWS commande.	10 mai 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateforme pour les `1.12` clusters Kubernetes afin de prendre en charge les noms DNS personnalisés dans le `kubelet` certificat et d'améliorer les performances. `etcd` Cela corrige un bogue qui faisait que les démons `kubelet` des nœuds redemandaient un nouveau certificat au bout de quelques secondes.	8 mai 2019

Didacticiel Prometheus	Ajout d'une rubrique pour le déploiement de Prometheus dans votre cluster Amazon EKS.	5 avril 2019
Journalisation de plan de contrôle d'Amazon EKS	Avec cette mise à jour, vous pouvez obtenir des journaux d'audit et de diagnostic directement à partir du volet de contrôle d'Amazon EKS. Vous pouvez utiliser ces CloudWatch journaux dans votre compte comme référence pour sécuriser et exécuter des clusters.	4 avril 2019
Version de Kubernetes `1.12`	Ajout de la prise en `1.12` charge des versions de Kubernetes pour les nouveaux clusters et les mises à niveau de version.	28 mars 2019
Ajout du guide de démarrage App Mesh	Ajout de la documentation pour démarrer avec App Mesh et Kubernetes.	27 mars 2019
Accès privé au point de terminaison du serveur d'API Amazon EKS	Ajout de documentation pour désactiver l'accès public pour le point de terminaison du serveur d'API Kubernetes de votre cluster Amazon EKS.	19 mars 2019
Ajout d'une rubrique pour l'installation du serveur de métriques Kubernetes	Le serveur de métriques Kubernetes est un agrégateur de données sur l'utilisation des ressources dans votre cluster.	18 mars 2019

Ajout d'une liste de projets open source connexes	Ces projets open source étendent les fonctionnalités des clusters Kubernetes exécutés sur ces clusters AWS, y compris les clusters gérés par Amazon EKS.	15 mars 2019
Ajout d'une rubrique pour l'installation de Helm en local	Le gestionnaire de package `helm` pour Kubernetes vous permet d'installer et de gérer des applications dans votre cluster Kubernetes. Cette rubrique explique comment installer et exécuter localemen t les fichiers binaires `helm` et `tiller`. Vous pouvez ainsi installer et gérer des graphiques à l'aide de la CLI Helm sur votre système local.	11 mars 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plate-forme qui met à jour les `1.11` clusters Amazon EKS Kubernetes au niveau des correctifs pour traiter le CVE-2019-10021001`.11.8`.	8 mars 2019
Augmentation de la limite de clusters	Amazon EKS a augmenté le nombre de clusters que vous pouvez créer dans une AWS région de 3 à 50.	13 février 2019

Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les AWS régions Europe (Londreseu-west-2) (), Europe (Pariseu-west-3) () et Asie-Pacifique (Mumbai) (ap-south-1).	13 février 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour ALAS-2019-1156	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS pour corriger la vulnérabilité décrite dans le document ALAS-2019-1156.	11 février 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour ALAS2-2019-1141	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS pour répondre aux problèmes mentionnés dans le CVEs document ALAS2-2019-1141.	9 janvier 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Séoulap-northeast-2) ().	9 janvier 2019
Extension de région Amazon EKS	Amazon EKS est désormais disponible dans les AWS régions supplémentaires suivantes : Europe (Francfor teu-central-1) (), Asie-Pacifique (Tokyoap-northeast-1) (), Asie-Pacifique (Singapour) ap-southeast-1 () et Asie-Pacifique (Sydney) ap-southeast-2 ().	19 décembre 2018

Mise à jour du cluster Amazon EKS	Ajout de documentation relative à la mise à jour de la version de Kubernetes des clusters et au remplacement de nœuds pour Amazon EKS.	12 décembre 2018
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Europe (Stockholm`eu-north-1`) ().	11 décembre 2018
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de la plateforme mettant Kubernetes à jour au niveau des correctifs pour corriger AWS link:security/security-bulletins/ `1.10.11` -2018-020 / [CVE-2018-1002105, type="marketing "].	4 décembre 2018
Ajout de la prise en charge de la version `1.0.0` du contrôleur d'entrée ALB	Le contrôleur d'entrée ALB publie une version `1.0.0` avec le support formel de. AWS	20 novembre 2018
Ajout de la prise en charge de la configuration réseau CNI	Le plug-in Amazon VPC CNI pour la version Kubernetes prend `1.2.1` désormais en charge la configuration réseau personnalisée pour les interfaces réseau Pod secondaires.	16 octobre 2018

Ajout de la prise en charge de `MutatingAdmissionWebhook` et `ValidatingAdmissionWebhook`	La version `1.10-eks.2` de la plateforme Amazon EKS prend désormais en charge les contrôleurs d'admission `MutatingAdmissionWebhook` et `ValidatingAdmissionWebhook`.	10 octobre 2018
Ajout d'informations d'AMI de partenaire	Canonical s'est associé à Amazon EKS pour créer un nœud AMIs que vous pouvez utiliser dans vos clusters.	3 octobre 2018
Instructions ajoutées pour la `update-kubeconfig` commande AWS CLI	Amazon EKS a ajouté le `update-kubeconfig` à la AWS CLI pour simplifier le processus de création d'un `kubeconfig` fichier permettant d'accéder à votre cluster.	21 septembre 2018
Nouvel Amazon EKS optimisé AMIs	Amazon EKS a mis à jour la version optimisée d'Amazon EKS AMIs (avec et sans prise en charge du GPU) afin de fournir divers correctifs de sécurité et optimisations des AMI.	13 septembre 2018
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la région Europe (Irlande) (`eu-west-1`).	5 septembre 2018

Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateform e avec support de la couche d'agrégation et Horizonta l Pod Autoscaler (HPA) de Kubernetes.	31 août 2018
Nouvel Amazon EKS optimisé AMIs et prise en charge du GPU	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS afin d'utiliser un nouveau modèle de AWS CloudForm ation nœud et un nouveau script d'amorçage. En outre, une nouvelle AMI optimisée pour Amazon EKS avec prise en charge GPU est disponible.	22 août 2018
Nouvelle AMI optimisée pour Amazon EKS corrigée pour `ALAS2-2018-1058`	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS pour répondre aux problèmes mentionnés dans le CVEs document ALAS2-2018-1058.	14 août 2018
Scripts de génération d'AMI optimisée pour Amazon EKS	Amazon EKS comporte des scripts de génération open source qui sont utilisés pour créer l'AMI optimisée pour Amazon EKS. Ces scripts de compilation sont désormais disponibles sur GitHub.	10 juillet 2018
Version initiale d'Amazon EKS	Documentation initiale pour le lancement de service	5 juin 2018

Contribuez au guide de l'utilisateur EKS

AWS a lancé une expérience de contribution améliorée pour le guide de l'utilisateur d'EKS.

Vous pouvez désormais modifier la source du guide de l'utilisateur EKS directement sur GitHub.

La documentation utilise AsciiDoc désormais un puissant langage de création similaire au markdown. AsciiDoc associe une syntaxe simple à des fonctionnalités de documentation d'entreprise telles que le formatage avancé, les références croisées et les contrôles de sécurité.

Vous pouvez désormais modifier les documents EKS directement sur GitHub. Notre processus simplifié inclut :

- Traitement plus rapide des pull requests
- Nombre réduit d'étapes manuelles
- Contrôles automatisés de la qualité du contenu

Nous attendons vos contributions avec impatience.

Modifier une seule page à partir d'un navigateur Web

Vous pouvez facilement modifier une seule page du guide de l'utilisateur EKS directement via votre navigateur Web.

Si vous souhaitez modifier plusieurs pages à partir de votre navigateur Web, consultezthe section called "Modifiez des fichiers avec GitHub".

Prérequis

- La page Docs à modifier s'est ouverte dans le navigateur Web.
- Connecté à GitHub.

Procédure

1. Accédez à la page que vous souhaitez modifier dans le guide de l'utilisateur Amazon EKS.
2. Dans le volet droit, choisissez le GitHub lien Modifier cette page sur.
3. Une fois activé GitHub, ouvrez l'éditeur de l'une des manières suivantes :
   - En appuyant sur la e touche de votre clavier.

- Cliquez sur l'icône en forme de crayon et sélectionnez Modifier sur place dans le menu déroulant.
- Si vous n'avez pas la possibilité de modifier, vous devez vous connecter à GitHub. Votre GitHub compte n'a pas besoin d'autorisations spéciales pour suggérer des modifications. Toutefois, les contributeurs internes d'Amazon doivent associer leur GitHub profil.

4. Apportez les modifications nécessaires au contenu dans l' GitHub éditeur.
   - L'éditeur fournit des fonctionnalités de surlignage syntaxique et de prévisualisation.
   - Vous pouvez utiliser le AsciiDoc balisage pour mettre en forme vos modifications.
   - Vous pouvez l'utiliser `ctrl-f` pour ouvrir une interface de recherche/remplacement.
5. (Facultatif) Prévisualisez vos modifications.
   - Utilisez l'`preview`onglet pour prévisualiser vos modifications grâce à une mise en forme enrichie.
   - Utilisez `show diff` cette option pour mettre en évidence les sections modifiées. Les sections supprimées ont un indicateur rouge dans la marge gauche. Les nouvelles sections ont un indicateur vert dans la marge gauche.
6. Lorsque vous avez terminé de modifier, cliquez sur le bouton Valider les modifications... en haut de l'éditeur
7. Dans la boîte de dialogue de validation :
   - Vérifiez que votre adresse e-mail est correcte.
   - Ajoutez un message de validation bref mais descriptif expliquant vos modifications.
   - Ajoutez éventuellement une description plus longue si nécessaire.
   - Sélectionnez cette option pour créer une nouvelle branche et une nouvelle pull request.

Vous avez créé une pull request incluant les modifications proposées.

Vue d'ensemble de la pull request

Lorsque vous créez un PR :

- Vos modifications sont soumises pour examen par les responsables du dépôt.
- Les réviseurs peuvent commenter vos modifications et demander des modifications.
- Des tests automatisés peuvent être exécutés pour valider vos modifications.
- Une fois approuvées, vos modifications peuvent être fusionnées dans le référentiel principal.

Les pull requests contribuent à garantir la qualité et permettent de discuter des changements avant qu'ils ne soient intégrés.

Découvrez comment les pull requests sont examinées et approuvées dans les GitHub documents.

Modifier plusieurs fichiers à partir d'un navigateur Web à l'aide de l'éditeur GitHub Web

Si vous souhaitez proposer la modification de plusieurs pages ou créer une nouvelle page de documentation, utilisez l'éditeur Web GitHub .dev. Cet éditeur Web est basé sur le célèbre éditeur de texte Visual Studio Code.

Prérequis

- Connecté à GitHub
- Connaissance de l'éditeur de code Visual Studio
- Connaissance de Git

Procédure

> **Note**
>
> L'équipe EKS Docs a créé un fichier d'espace de travail qui inclut des suggestions de configurations pour l'éditeur, telles que l'habillage du texte et le surlignage AsciiDoc syntaxique. Nous vous conseillons de charger ce fichier d'espace de travail.

1. Ouvrez l'espace de travail sur GitHub .dev.
   - Vous pouvez ajouter l'URL à vos favoris `https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true`
2. (Première configuration uniquement) Vous serez peut-être invité à créer un fork du dépôt dans votre propre GitHub compte. Acceptez cette invite. Pour plus d'informations, consultez la section À propos des fourchettes dans la GitHub documentation.
3. (Première installation uniquement) Acceptez l'invite en bas à droite pour installer l' AsciiDoc extension.
4. Accédez au contenu de la documentation à l'adresse`latest/ug`.
   - Les fichiers Docs sont organisés par section de niveau supérieur. Par exemple, les pages du chapitre « Sécurité » contiennent des fichiers source dans le répertoire « security/ ».
5. Pour afficher un aperçu d'une page de documentation, utilisez le bouton Ouvrir l'aperçu sur le côté en haut à droite. L'icône inclut une petite loupe.
6. Utilisez l'onglet Contrôle de source sur la gauche pour valider vos modifications. Pour plus d'informations, consultez la documentation de Visual Studio Code :
   - Valider les modifications
   - Créer une demande d'extraction

Une fois que vous avez créé une pull request, elle sera examinée par l'équipe de documentation.

Consultez les commentaires sur le style au fur et à mesure que vous tapez en installant Vale localement

Vous pouvez voir les commentaires relatifs au style au fur et à mesure que vous tapez. Cela permet d'identifier les fautes d'écriture et les fautes de frappe.

Vue d'ensemble :

- La CLI Vale charge les guides de style et les exécute par rapport aux fichiers source.
- Le dépôt EKS Docs inclut un fichier de configuration de valeur qui charge les guides de style et les règles locales.
- L'extension Vale pour Visual Studio (VS) Code affiche les informations relatives aux valeurs dans l'éditeur.

Installez Vale

Suivez les instructions de la documentation de la CLI Vale pour installer Vale avec un gestionnaire de packages.

Installez l'extension VS Code Vale

1. Ouvrez VS Code.

2. Cliquez sur l'icône Extensions dans la barre d'activité (ou appuyez sur Ctrl+Shift+X).
3. Recherchez « Vale ».
4. Cliquez sur Installer sur l'extension « Vale VSCode » de Chris Chinchilla.
5. Rechargez VS Code lorsque vous y êtes invité.

Valeur de synchronisation

Vale utilise le fichier `.vale.ini` de configuration à la racine de votre projet pour déterminer les règles de style à appliquer.

1. Ouvrez VS Code.
2. Cliquez sur Afficher **>** Terminal (ou appuyez sur Ctrl+`).
3. Accédez au répertoire racine de votre projet si nécessaire.
4. Exécutez la commande :

```
vale sync
```

5. Attendez que Vale ait fini de télécharger et de synchroniser les règles de style

Afficher les commentaires de style dans VS Code

1. Ouvrez un Markdown ou un AsciiDoc fichier dans VS Code.
2. L'extension Vale vérifiera automatiquement votre texte par rapport aux règles de style.
3. Les problèmes de style seront soulignés dans l'éditeur.
4. Passez le curseur sur le texte souligné pour voir la suggestion de style spécifique.
5. Résolvez les problèmes en suivant les suggestions ou en consultant le guide de style.

Créez une nouvelle page

Découvrez comment créer une nouvelle page de documentation. Cette rubrique contient des instructions pour créer les métadonnées de page initiales et ajouter la page à la table des matières du guide.

Créer une page

1. Accédez au répertoire des chapitres. Par exemple, si vous souhaitez créer une nouvelle page dans la section « Sécurité », accédez au `latest/ug/security` répertoire.
2. Déterminez l'ID de page. Par convention, l'ID de page est entièrement en minuscules et segmenté avec. `-` L'ID de cette page est`create-page`.
3. Créez un nouveau fichier avec l'ID de page et l'`adoc`extension. Par exemple, `create-page.adoc`.
4. Insérez les métadonnées de page à l'aide de ce modèle :

```
include::../attributes.txt[]

[.topic]
[#unique-page-id]
= Page title
:info_titleabbrev: Short title

[abstract]
--
Brief summary of the page's content.
--

Introduction paragraph goes here.
```

Ajouter une page à la navigation

1. Accédez à la page parent. La page parent des sections de niveau supérieur est`book.adoc`.
2. Au bas de la page parent, incluez la page enfant.

   ```
   include::${filename}[leveloffset=+1]
   ```

   Par exemple :

   ```
   include::create-page.adoc[leveloffset=+1]
   ```

Insérer un lien

AsciiDoc prend en charge plusieurs types de liens. Il est important d'utiliser le bon type de lien pour que le lien fonctionne correctement dans différents environnements.

Lien vers une page ou une section du guide de l'utilisateur EKS

Utilisez des références croisées (xref) pour établir des liens entre les pages ou les sections d'un même site de documentation, tel que le guide de l'utilisateur EKS. Ils sont automatiquement mis à jour si la section cible est déplacée ou renommée.

Utiliser le titre de page comme texte de lien

Dans la plupart des cas, lorsque vous créez un lien vers un autre identifiant dans ce guide de l'utilisateur, utilisez l'approche suivante pour que le texte du lien soit automatiquement mis à jour avec le dernier titre selon les besoins.

```
For more information, see <<page-or-section-id>>.
```

Définir un texte de lien personnalisé

Dans les cas où vous devez avoir un texte de lien personnalisé, utilisez le format suivant.

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

Lien vers un autre guide dans les AWS documents

1. Trouvez le lien vers la page de AWS documentation.
2. Supprimez le `https://docs.aws.amazon.com/` préfixe en ne conservant que le chemin. Le chemin doit commencer par une lettre.
3. Créez un lien comme indiqué ci-dessous :

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,
 type="documentation"]
```

Lien vers une page Web externe

Ce format crée un lien standard vers une page non hébergée par Amazon. Par exemple, utilisez-le pour les GitHub liens.

```
https://example.com[Link text]
```

> **Note**
>
> Nous avons une liste d'autorisations pour les domaines externes. La liste des autorisations se trouve à `vale/styles/EksDocs/ExternalDomains.yml`

Création de contenu documentaire avec Amazon Q

Vous pouvez utiliser Amazon Q pour créer et réviser le contenu des documents. Il s'agit d'un moyen simple de démarrer sur une nouvelle page. Amazon Q est disponible en tant qu'extension de Visual Studio (VS) Code.

Dans l'image suivante, Amazon Q a généré les lignes marquées en vert.

Installez Amazon Q avec VS Code

1. Ouvrez VS Code
2. Accédez à la vue Extensions (Ctrl+Shift+X ou Cmd+Shift+X)
3. Recherchez « Amazon Q »
4. Cliquez sur Installer sur l'extension Amazon Q
5. Attendez que l'installation soit terminée
6. Redémarrez VS Code lorsque vous y êtes invité

Connectez-vous à Amazon Q

1. Après l'installation, choisissez l'icône Amazon Q dans la barre d'activité de VS Code.
2. Choisissez Se connecter à Amazon Q.
3. Entrez vos AWS informations d'identification lorsque vous y êtes invité.

4. Une fois authentifié, vous verrez l'interface de chat Amazon Q.

Utiliser Amazon Q pour créer du contenu

1. Ouvrez le fichier que vous souhaitez modifier dans VS Code.
2. Sélectionnez le texte que vous souhaitez réviser ou l'emplacement du nouveau contenu.
3. Appuyez sur Ctrl+I ou Cmd+I.
4. Dans l'invite, soyez précis sur les points suivants :
   - Le type de contenu dont vous avez besoin.
   - Le public cible.
   - Principaux points à aborder.
   - Ton et style souhaités.
5. Passez en revue le contenu généré dans l'aperçu en ligne.
6. Utilisez enter pour accepter les modifications ou esc pour les rejeter.
7. Modifiez davantage si nécessaire.

Conseils

- Commencez par une simple demande et répétez pour obtenir le contenu souhaité.
- Créez une première ébauche des en-têtes de page, puis demandez à Q de les remplir.
- Amazon Q peut générer du Markdown. C'est très bien. L' AsciiDoc outillage peut comprendre la plupart des syntaxes du Markdown.

Pour en savoir plus sur Amazon Q Developer, consultez la section Utilisation d'Amazon Q Developer dans l'IDE.

Afficher un aperçu du contenu de la pull request

Le guide de l'utilisateur Amazon EKS GitHub est configuré pour créer et générer un aperçu du site de documentation. Cet aperçu ne contient pas le AWS thème complet, mais il vérifie que le contenu est correctement créé et que les liens fonctionnent.

Cet aperçu est hébergé sur une URL temporaire par AWS Amplify.

Afficher l'aperçu

Lorsque vous soumettez une pull request, AWS Amplify tente de créer et de déployer un aperçu du contenu.

Si la compilation réussit, aws-amplify-us-east-1 ajoute un commentaire à la pull request contenant un lien vers l'aperçu. Cliquez sur le lien situé à droite de « Accédez à cette pull request ici » (comme indiqué sur la capture d'écran avec un contour rouge).

Si la compilation échoue, les administrateurs du dépôt peuvent consulter les journaux et fournir des commentaires.

> **Note**
>
> Si vous n'avez jamais contribué auparavant, un responsable du projet devra peut-être approuver l'exécution de la compilation.

Limitations propres à la version préliminaire

L'aperçu est créé sous la forme d'un seul gros fichier HTML. Il sera affiché sous forme de plusieurs pages lors de sa publication.

Ce qui fonctionne :

- Références croisées (`xref`)
- Liens vers Internet
- Images
- Contenu hébergé par `samples/`

Ce qui ne fonctionne pas :

- Liens vers d'autres AWS contenus, en utilisant`type="documentation"`. Cela est dû au fait que ce contenu n'existe pas dans l'environnement de prévisualisation.
- L'attribut ne `{aws}` s'affichera pas correctement. La valeur de ce paramètre change en fonction de l'environnement.

AsciiDoc référence de syntaxe

AsciiDoc est le langage de balisage préféré pour la AWS documentation. Bien que l'outil supporte partiellement la syntaxe Markdown (comme les en-têtes et les listes), nous recommandons d'utiliser la AsciiDoc syntaxe pour l'ensemble du contenu afin de garantir la cohérence et un rendu correct.

Ce guide couvre les éléments de syntaxe courants dont vous aurez besoin pour contribuer à la documentation Amazon EKS. Pour une syntaxe plus avancée et des informations détaillées, reportez-vous à la AsciiDoc documentation.

Nouvelle page

Chaque nouveau AsciiDoc document doit commencer par la structure définie dansthe section called "Créer une page".

En-têtes

Utilisez des titres pour organiser votre contenu de manière hiérarchique :

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
====== Level 6 heading
```

Remarque : utilisez toujours des majuscules pour les titres de la AWS documentation.

Formatage du texte

Formatez le texte pour mettre en valeur les informations importantes :

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

Listes

Listes non ordonnées

Créez des listes à puces pour les éléments sans séquence spécifique :

```
* First item
* Second item
** Nested item
** Another nested item
* Third item
```

Listes ordonnées

Créez des listes numérotées pour les étapes séquentielles ou les éléments prioritaires :

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

Liens

Consultez the section called "Insérer un lien" pour plus de détails sur la manière de formater correctement les liens. Les liens de style Markdown ne sont pas pris en charge.

Exemples de code

Code en ligne

Utilisez des backticks pour le code en ligne :

```
Use the `kubectl get pods` command to list all pods.
```

Blocs de code

Créez des blocs de code avec surlignage syntaxique et prise en charge des attributs (similaires aux entités) :

```
[source,python,subs="verbatim,attributes"]
----
def hello_world():
    print("Hello, World!")
----
```

Images

Insérez des images avec du texte alternatif pour des raisons d'accessibilité :

```
image::images/image-file.png[Alt text description]
```

Tables

Créez des tableaux pour organiser les informations :

```
[%header,cols="2"]
|===
|Header 1
|Header 2

|Cell 1,1
|Cell 1,2

|Cell 2,1
|Cell 2,2
|===
```

Pour des tables plus complexes, consultez la documentation relative aux AsciiDoc tables.

Alertes

Utilisez des légendes pour mettre en évidence les informations et les avertissements importants :

```
NOTE: This is a note callout for general information.

TIP: This is a tip callout for helpful advice.

IMPORTANT: This is an important callout for critical information.
```

Aperçu :

> **Note**
>
> Il s'agit d'une légende.

Inclure d'autres fichiers

Incluez le contenu d'autres fichiers :

```
include::filename.adoc[]
```

Attributs (similaires aux entités)

Utilisez des attributs prédéfinis pour garantir la cohérence. En particulier, vous DEVEZ utiliser des attributs pour AWS et `arn:aws:` .

```
{aws} provides Amazon EKS as a managed Kubernetes service.
```

```
[source,bash,subs="verbatim,attributes"]
----
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn {arn-aws}iam::aws:policy/AmazonEKSClusterPolicy
----
```

Pour obtenir la liste des attributs, consultez le `../attributes.txt` fichier.

Procédures

step-by-stepProcédures de formatage :

```
To create an Amaozon EKS cluster. do the following steps.

. Sign in to the {aws} Management Console.
. Open the Amazon EKS console.
. Choose *Create cluster*.

...
```